

Németh József

Adat- védelem



a számítógépes
és hírközlő
rendszerekben

**Adatvédelem
számítógépes és hírközlő rendszerekben**

**Data protection
in computer and telecommunications systems**

József Németh

Data protection in computer and telecommunications systems

**Computing Applications and Service Company
Budapest, 1984**

Németh József

Adatvédelem számítógépes és hírközlő rendszerekben

**Számítástechnika-alkalmazási Vállalat
Budapest, 1984**

Lektorálta: dr. Borda József

Supervised by dr. József Borda

© Németh József, 1984

ISBN 963 553 054 4

Tartalom

Előszó	7
1. Bevezetés	9
1.1 Számítógépes bűnözés	9
1.2 Az adatvédelem szerkezete	11
2. Rejtjelezés	15
2.1 Alapfogalmak	15
2.2 Rendszertechnikai problémák	19
2.3 Néhány hagyományos rejtjelezési eljárás	22
2.4 Bitrejtés	26
3. Blokkrejtés	29
4. Az IBM rejtjelező rendszerei	35
4.1 DES (Data Encryption Standard)	36
4.2 LUCIFER	45
4.3 A DES értékelése	47
5. Nyilvános kulcsú rejtés	53
5.1 Az MIT eljárás	54
5.2 A Merkle–Hellman módszer	57
6. Kulcsgondozás	65
6.1 Kulcskiosztás	65
6.1.1 Merkle „rejtvény” módszere	65
6.1.2 Nyilvános kulcsú kiosztás	66
6.1.3 Alapkulcsrendszer	67
6.2 Kulcselőállítás	73
6.3 Kulcstárolás	77
7. Kapcsolathitelesítés	81

7.1	Üzenethitelesítés	82
7.1.1	Üzenetláncolás	82
7.1.2	Blokkláncolás	85
7.1.3	Hibakorlátozás	87
7.2	Digitális kézjegy	87
7.2.1	Közvetített digitális kézjegy	89
7.2.2	Valódi digitális kézjegy	92
7.2.3	Egyéb digitális aláírási módszerek	94
7.3	Partnerazonosítás	97
7.3.1	Kapcsolatfelvétel konvencionális rejtjelezési eljárást alkalmazó rendszerekben	97
7.3.2	Kapcsolatfelvétel nyilvános kulcsú rejtjelezési eljárást alkalmazó rendszerekben	99
7.4	Egyirányú kapcsolat	100
8.	Hozzáférés-védelem	103
8.1	Az adathozzáférés szabályozásának rendszermodellje	104
8.2	A hozzáférési mátrix tárolása	109
8.2.1	Engedélyek	109
8.2.2	Felhatalmazások	111
9.	Hozzáférés-védelem gyakorlati formái (példái)	113
9.1	Virtuális tár	113
9.2	Háttértár-védelem	119
9.3	A hozzáférési lista szerkezete	121
9.4	Védelmi alrendszerek	126
9.5	Az engedély-visszavonás módszerei	127
9.5.1	Engedélyek tárolása	128
9.5.2	Az engedély-visszavonás változatai	131
9.5.3	Engedélyterjesztési gráf	135
9.5.4	Kapcsolt engedélyek	137
10.	Felhasználóazonosítás	143
10.1	Jelszóvédelem rejtjelezéssel	146
10.2	Jelszóvédelem jelszó sorozattal	152

Contents

1. Introduction
2. Cryptography
3. Block encryption
4. The IBM cryptographic system
5. Public-key cryptography
6. Key maintenance
7. Authentication
8. Access control
9. Practical forms and examples of access control
10. User identification

Előszó

Az adatvédelem célja és feladata a számítógépes rendszerek rendeltetésszerű felhasználásának szavatolása. Központi jelentősége annak a felismerésnek a következménye, hogy a számítógépek alkalmazása a kétségtelen előnyei mellett társadalmi méretű problémák okozójává válhat. Sőt mi több, ezeknek a problémáknak a forrásai ugyanazok az új lehetőségek, amelyek a számítógépes rendszerek rohamos terjedését nemcsak elősegítik, hanem elkerülhetetlenné is teszik.

Számítógépes rendszerekben hatalmas mennyiségű adat (és ezzel egyidejűleg információ) tárolható, nemhogy napra-, de „pillanatra kész” állapotban; az adatok tetszőleges feltételek szerinti visszakeresése, a tárolt adatok belső kapcsolatainak megállapítása percek műve; a nagy sebességű adatközlő csatornák bekapcsolásával pedig az egész világot átfogó rendszerek hozhatók létre. A számítógépek teljesítőképességének ismeretében nem kell elszabadult fantázia például annak felismeréséhez, hogy egyedi közigazgatási, egészségügyi, vállalati, helyfoglalási, utazási irodai rendszerek és információs közművek (mint pl. a számítógépesített telefonközpontok, kábeles televíziós hálózatok) összekapcsolásával olyan rendszerek hozhatók létre, amelyekkel az egyén életének minden pillanata születésétől a haláláig nyomon követhető, mikroszkóp alá vehető. És ahhoz sem kell született pesszimistának vagy elvakult individualistának lennünk, hogy egy ilyen rendszer kialakulását és működésbe lépését egy lehetséges nukleáris háborúval azonos értékű fenyegetésnek tartsuk.

Az adatvédelemnek tehát fontos eleme a jogi szabályozás, amely a számítógépes rendszerek nyújtotta lehetőségek és társadalmi szintű igények harmonikus összehangolását hivatott megalapozni.

Az adatvédelmet szolgáló jogszabályok kidolgozása bonyolult és lassú folyamat; az adatok és egyének viszonyát, az adatokat felhasználó személyek és szervezetek jogait és kötelességét, az adatáramoltatás módjait és megszorításait kell minden részletére kiterjedően és a kialakult törvénykezési rendszer kereteibe illően kidolgozni. Ez a munka világ-méreteken is csak a kezdet kezdeténél tart, bár a számítástechnika alkalmazásában érzékelhető fejlettségi szintet elért országok többségében az ide vonatkozó legáltalánosabb szabályozás már életbe lépett. (Hazánkban az 1/1981(I.27.)BM sz. rendelet intézkedik a számítástechnikai rendszerek titok-, vagyon- és tűzvédelmének alapfeladatairól.)

A számítógépek alkalmazásának egy másik potenciális katasztrófaforrását az jelenti, hogy egyes szolgáltatásokban a számítógépes rendszerek térhódítása már ma is totális, és egyre inkább azzá válik minden lényeges szolgáltató rendszerben. Az adatvédelem feladata tehát az is, hogy a számítógépes szolgáltató rendszerek megbízható működését a hagyományos rendszerekével azonos szinten biztosítsa. Ennek szükségességét és elkerülhetetlenségét

ismerték fel legkorábban, ezért ma már a rendszer megbízhatóság kívánt értékének eléréséhez minden műszaki területre és részletre kiterjedő megoldásaink vannak. A megoldások közös eleme a berendezésszintű, működésbeli és rendszertechnikai redundancia alkalmazása.

Az adatvédelem harmadik feladatát a számítástechnikai rendszerek biztonságának megteremtése jelenti. Másképpen megfogalmazva: annak a rendszernek a kiépítése, amelynek segítségével a jogi szabályozás kötelező erővel érvényre juttatható.

A témával foglalkozó szakirodalomban a védelmi szintek kiépítésével kapcsolatban számos fogalom, csoportosítási el ismert. Vannak, akik az eszközrendszer jellege szerint csoportosítják a védelmi eszközöket, míg mások aszerint, hogy az adott védelmi szinten milyen körben érvényesíthetik hatásukat.

A biztonságtechnikai rendszereket a következő csoportosításban tárgyaljuk:

Fizikai védelem: a számítógépes rendszer védelmének „hagyományos” eszközeit egyesíti (mint pl. biztonságos elhelyezés, biztonságosan lezárt berendezéskialakítás stb.)

Ügyviteli védelem: a rendszert üzemeltető személyzet szabályozott tevékenységével megvalósított védelmi réteget képezi. Elemei: személyes felelősségre vonás, az egyéni felelősség megállapíthatósága, a hozzáférés és a tevékenységi körök korlátozása.

Algoritmikus védelem: azoknak a védelmi eszközöknek az együttese és rendszere, amelyek védelmi feladatukat a rendszerbe építve és a szolgáltatások megvalósításával együttműködve látják el.

Könyvemben az adatvédelem bonyolult és, mint látható, szerteágazó rendszeréből csak ezt a legutolsóként említett elemet, az algoritmikus védelmet választottam ki ismertetésre.

1. Bevezetés

1.1 Számítógépes bűnözés

A számítógépes rendszerekkel elkövetett visszaélések, visszaélési kísérletek egyidősek magával a számítógéppel. Az első számítógépes rendszerek korlátozott alkalmazási köre még csak „ártalmatlan” visszaélésekre adott lehetőséget: ingyenes gépidő jogtalan igényénybevétele, egy zárt közösségben mások programjának felhatalmazás nélküli használata stb. A számítógépre alapozott szolgáltató rendszerek térhódításával – elsősorban a bankügyi rendszerek megjelenésével – párhuzamosan egy-egy sikeres visszaélés egyre nagyobb „zsákmányt” jelentett az elkövető **behatolónak**. A nagyobb haszon reménye pedig a számítógépes rendszer illegális és rendeltetésellenes használatba vételéhez utat nyitó **behatolás** formáinak és módszereinek kidolgozására ösztönzött. Az eredmény: a visszaélések számának robbanásszerű növekedése, a „számítógépes bűnözés” kialakulása.

A „számítógépes bűnöző” elsődleges célja természetesen pénz formájában a közvetlen anyagi haszon. Néhány példa az Egyesült Államokban feldolgozott esetek közül:

- Egy raktári tisztviselő leltárt hamisított, ezzel leplezve a tettestársához irányított áruszállításokat. A kár 4 millió dollár.
- Egy banktisztviselő átutalásokat hamisított. Kár: 1,4 millió dollár.
- Egy könyvelő fiktív neveket illesztve egy vállalat alkalmazotti listájához elérte, hogy bérüket a rendszer a saját számlájára utalja át. A kár 100 ezer dollár.
- Egy biztosítási ügynök kártérítési összegeket utalt át tettestársainak valótlán követelések kifizetésének engedélyezésével. A kár 128 ezer dollár.
- Egy számítógép-kezelő 22 db piaci információkat tartalmazó mágnesszalagot tulajdonított el, és csak 2000 dollár ellenében szolgáltatotta vissza őket.

A behatolók kedvenc célpontjai a bankügyleteket bonyolító rendszerek, mert ezek sikeres támadás esetén a legközvetlenebb formában szolgáltatják az eredményt. Több millió dollárt kitevő összeget tulajdonított például el egy bank számítógépes rendszerének programozója azzal, hogy az ügyletek során alkalmazott kerekítési szabályokat megváltoztatva ügyletenként jelentéktelen (néhány centnyi) összeget saját számlájára utaltatott át. Üzelmeit kollégái leplezték le, a bank szokásos ellenőrző rendszere nem mutatta ki. A közvetlen pénzszerzés mellett bizalmas információk, adatok, programok ingyenes megszerzése szerepel a visszaélések gyakori céljaként. Néhány „egzotikusabb”, számítógép segítségével eltulajdonított értéktárgy, szintén az egyesült államokbeli esetek listájáról:

- 217 vasúti teherszállító vagon;
- 60 millió gallon (228 millió liter) benzin;
- katonai egyenruhák;
- 1 millió dollár értékű telefonkészülék és telefonközponti tartalék alkatrész;
- 1000 vezetői jogosítvány (ez az USA-ban a személyi igazolvány szerepét is betölti);

- 3 millió nevet tartalmazó címlista (az USA-ban egy-egy címmel ellátott név a hirdetősi, reklámozási, postai úton történő vásárlási rendszer fejlettsége miatt jelentős értéket képvisel);
- Szavazatok (szavazásra nem jogosultak választói jegyzékbe való felvételével);
- Kedvező hitelképesség igazolása (amely jelentős pénzügyi csalásokat alapozhat meg).

A feldolgozott esetek figyelemre méltó adata, hogy míg egy átlagos fegyveres bankrablással eltulajdonított pénz az USA-ban 1978-ban 10 000 dollár volt, egy számítógépes visszaélés eredménye átlagosan 430 000 dollár.

A számítógépes bűnözés terjedésével a behatolás elleni védekezés eszközeinek, módszereinek, rendszertechnikájának kidolgozása kulcsfontosságúvá vált. A sikeres visszaélések esettanulmányai pedig rámutattak a hagyományos számítógépes rendszerek védelmének hiányosságaira, sebezhető pontjaira. A 60-as évek második felében megindult intenzív kutatómunka egy évtized alatt kialakította és részletekbe menően vizsgálta a számítógépes rendszeren belüli védelem struktúráját és eszköztárát. A kutatás eredményei a 70-es évek közepén kezdtek megjelenni a kereskedelmi forgalomban beszerezhető számítógépek hardver- és szoftverkomponenseiben. Időközben a rendszertervezők két olyan tényezővel is szembe találták magukat, amelyek a késztermék szintű védelmi rendszerek megjelenését késleltették.

Elsősorban az vált világossá, hogy egy számítógépes rendszerbe megfelelő szintű védelmet utólag „beépíteni” lehetetlen, a védelem követelményeit a teljes rendszer tervezésének már a legkorábbi fázisában (tehát már a berendezések tervezésekor) figyelembe kell venni. Ez még egy gyártó cég egyébként már kidolgozott védelmi rendszerének megjelenését is legalább egy következő számítógépmodell forgalomba hozataláig késlelteti. Továbbá növeli a késleltetést, hogy a védelmi rendszert szigorú kompatibilitási követelmények betartásával kell és csak jelentős szoftverátdolgozás árán lehet a teljes rendszerbe beépíteni.

A másik alapvető felismerésnek az bizonyult, hogy a védelmi rendszerekről — legalábbis jelenlegi eszközeinkkel — matematikai szigorúsággal általában nem bizonyítható be, hogy adott feltételek (amelyeket a számítógépes rendszereket körülvevő ügymenet teremt meg) mellett „hézagmentes” védelmet nyújtanak. Egy védelmi rendszert mindmáig csak behatolási kísérletekkel tudunk minősíteni. Erre a célra általában olyan behatoló csoportokat hoznak létre, akik a védelmi rendszert és annak megvalósítását minden részletében ismerik. Ezek azután ismereteikre támaszkodva minden lehetséges, rendelkezésükre álló eszközt igénybe véve igyekeznek a védelmi rendszer „mögé kerülni”. Természetesen csak az egyáltalán elérhető határfelületeken keresztül.)

Ez a „bizonyítási” módszer azonban hosszadalmas, csak a fejlesztési fázis után alkalmazható, és megkívánhatja a rendszer folyamatos átdolgozását.

A védelmi rendszer kidolgozása tehát időigényes feladat. Mindezt jól érzékelteti a tényleges fejlődés menete is: a számítógépes rendszerek védelmi rendszerének kialakításával kísérleti körülmények között már a 60-as évek elején elkezdtek foglalkozni, de a világon jelenleg üzemelő számítógépes rendszerek zöme még mindennemű védelmi eszközt nélkülöz, és a védelemmel rendelkező rendszerek is csak felületes behatolási kísérleteknek képesek ellenállni.

1.2 Az adatvédelem szerkezete

Számítógépes rendszerekben az adatvédelemnek három különböző formáját különböztetjük meg. Ezeket egymás közti viszonyuk alapján a védelem szintjeinek is nevezzük.

A **fizikai védelem** a berendezésbe vezető hozzáférési utak belépési pontjainak számát korlátozza. A számítógépet és közvetlen perifériáit (mágnesszalag, mágneslemez, operátori konzol stb.) általában betörésbiztos helyiségbe teszik. Ebbe csak néhány (legtöbbször egyetlen) ajtón keresztül lehet belépni, a behatolás egyéb lehetséges pontjait (pl.: világító és szellőzőnyílások) riasztóberendezések biztosítják. Sok esetben magukon a hivatalos belépési pontokon is csak védelmi berendezések felügyelete (mint pl.: kód- vagy azonosítókártyás zár) mellett lehet bejutni.

Az ellenálló felépítéssel, illetve elzárással szemben a fizikai védelem passzív formáját képviselik az önmegsemmisítő szerkezetek. Ezeknek a hozzáférési illetékesség szempontjából érzékeny adatokat, információkat tartalmazó részük automatikusan megsemmisül, ha a berendezést felbontják.

Az önmegsemmisítés leggyakrabban a berendezésben pillanatnyilag tárolt érzékeny adatok törléséből áll.

Az **ügyviteli védelem** a rendszert üzemeltető szervezet ügymenetébe épített biztonságtechnikai szabályok, viselkedések együttesét öleli fel. Gyökerénél mindig az egyéni felelősség és felelősségre vonás elve lelhető fel. Módszere a rendszer üzemelésének olyan szervezési és adminisztratív nyomon követése, amely egyrészt a behatolás, szabályellenes tevékenység lehető legkorábbi felfedését, másrészt ezek megszüntetésekor a felelősség megállapítását lehetővé teszi. A procedurális védelem általában két elemből építkezik:

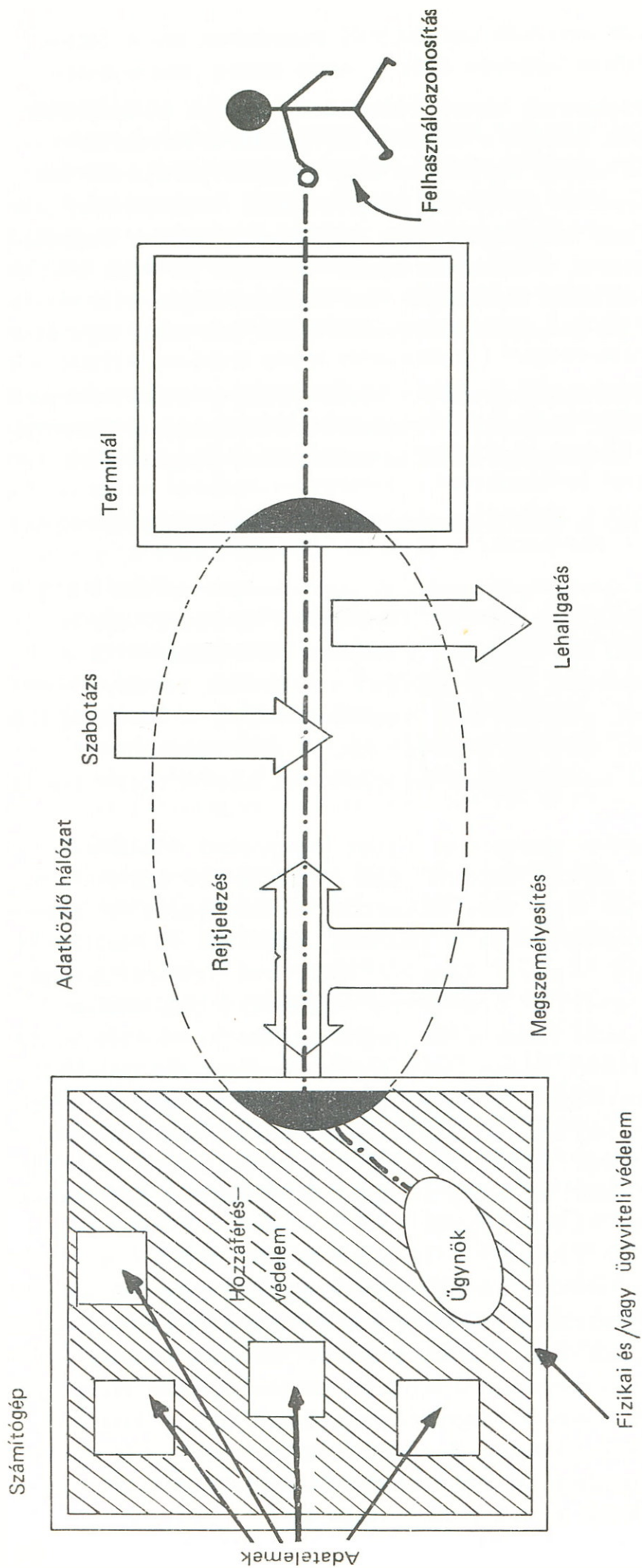
- a védelem szempontjából lényeges tevékenységek felügyeletéért kijelölt személyek felelősek,
- az ilyen tevékenységekről megfelelően részletes és hiteles feljegyzések készülnek.

A teljes ellenőrzés elég összetett ahhoz, hogy ne csak magányos behatolókat, hanem összejátszó alkalmazottakat is tetten érjen vagy leleplezzen. Számítógépes rendszerekben az ügyviteli védelem részeként a tevékenységekről részletes, időpontra és hozzáférési pontra való hivatkozásokat is tartalmazó feljegyzés (angolul: audit trail, szakmai szargonban „log”) készül, amely azután összevethető az üzemeltetés megfelelő feljegyzéseivel.

Az ügyviteli védelem a teljes rendszer védelmének meghatározóan fontos eleme, mivel a számítógépes rendszerek üzemvitelének jellege miatt számos tevékenység (operátori beavatkozások, adathordozók — mágnesszalagok, lemezcsomagok stb. — mozgatása, rendszerkarbantartás) más eszközökkel nem védhető. Az ügyviteli védelem a fizikai védelemre épül, a teljes védelem egy következő rétegét képezi. Amíg a fizikai védelem az engedélyezett rendszerbe való belépési pontokat jelöli ki, addig az ügyviteli védelem a belépési pontok igénybevételének elfogadható, elvárt formáit rögzíti.

Az **algoritmikus védelem** a védelem harmadik — az előző kettőre épülő — eleme. Azokból az eljárásokból áll, amelyek a rendszer szolgáltatásával egyidejűleg, velük szorosan együttműködve látják el védelmi feladataikat. A könyvben a védelemnek csak ezzel a lépcsőjével foglalkozunk, a fizikai és ügyviteli védelem jelenlétét azonban mindig feltételezzük. Magát az algoritmikus védelmet megvalósító rendszer például számos olyan építőelemet tartalmaz, amelyeknek megbízhatóan és meghamisíthatatlanul kell működniük, de erre az algoritmikus védelem már nem alkalmas. A rendszer-karbantartás programeszközei is csak az ügyviteli védelem felügyelete mellett tekinthetők megbízhatóknak.

Az algoritmikus védelem fő területeit az 1.1 ábra mutatja be.



1.1 ábra.

A számítógépes rendszer felhasználója a számítástechnikai rendszeren kívül álló elem, az algoritmikus védelem hatáskörébe tehát nem vonható be. A rendszer biztonsága érdekében a felhasználó kilétének hitelt érdemlő ismerete ugyanakkor létfontosságú. Az algoritmikus védelemnek ezért a **felhasználóazonosítás** az egyik fontos területe. Az alapelve igen egyszerű: a felhasználóról a rendszerben tárolt adatokat, információt összehasonlítjuk azokkal, amelyeket a felhasználó a rendszerbe való belépéskor megad. Az azonosító adatok, információk a rendszerbe az ügyviteli védelem felügyelete mellett kerülnek, és feltételezzük, hogy titkosak, a felhasználóra jellemzően egyediek. A felhasználóazonosítás tehát védtelen az ellen, ha a felhasználó „megszemélyesítése” (kilétének kisajátítása) teljes egészében a rendszeren kívül történik.

Az algoritmikus védelem feladata azonban, hogy az azonosító adatok, információk titkosságát és hitelességét érvényességük időtartamára biztosítsa, továbbá megakadályozza a behatolót abban, hogy ezeket a bejelentkezés folyamatába való behatolással megszerezze.

A felhasználóazonosítás módszereit a könyv legutolsó, 10. fejezetében ismertetjük, mivel ez a másik két nagy algoritmikus védelmi terület (a rejtjelezés és a hozzáférés-védelem) módszereit, eszközeit jelentős mértékben felhasználja.

Rejtjelezés

Az adatközlő hálózat a teljes rendszer legsebezhetőbb eleme: földrajzi kiterjedésénél fogva ügyviteli védelme megszervezhetetlen, fizikai védelme elviselhetetlenül költséges volna. Az adatközlő utakban továbbított adatokhoz a behatoló tehát lehallgatással minden különösebb nehézség nélkül hozzájuthat. Az egyetlen lehetséges védelmet az adatok továbbítás előtti olyan átalakítása jelenti, amely a behatoló számára felhasználhatatlanná, érthetlenné teszi őket. A teljes eljárás, a **rejtjelezés**, két leképezést: a feladó alkalmazta rejtést és a címzett birtokában lévő fejtést foglalja magában. A védő hatás abban van, hogy a rejtéshez valamilyen információ, az ún. **kulcs** ismerete szükséges, amelyet a címzett titokban tart. **Konvencionális rejtjelezési módszerekben** ugyanez a kulcs a megfejtésben is szerepet játszik, tehát a feladónak is titokban kell tartania. **Nyílt rejtőkulcsú rendszerekben** a rejtés teljes ismeretében sem lehet a megfejtés kulcsára következtetni. Mindkét típusú rejtjelezés lényege azonban ugyanaz: a behatoló nem lévén birtokában a fejtőkulcsnak, a rejtett adatokat nem képes megismerni, belőlük még a rejtetlen, ún. nyílt formájuk ismeretében sem tud a fejtőkulcsra következtetni.

A **lehallgatás** passzív behatolási forma, a behatolót azonban semmi sem akadályozza abban, hogy aktívan beavatkozzon az adatkapcsolatba. Ezeket a lehetőségeket szintén az 1.1 ábrán tüntettük fel.

A legegyszerűbb aktív behatolási forma a **szabotázs**: a behatoló célja az, hogy megakadályozza az információcserét az adatkapcsolatban. A legnagyobb kárt nyilvánvalóan, akkor okozza, ha ezt észrevétlenül, az adatkapcsolaton áramló információk meghamisításával teszi. A szabotázhhoz a behatolónak nem kell megismernie az áramló adatokat! Célját eléri, ha megakadályozza, hogy az adatok egy része eljusson a címzethez, vagy ha egy korábbi kapcsolatból kivont rejtjelezett adatokat illeszt be az adatfolyamba. Kevésbé „kifinomult” módszer a rejtett adatok továbbítás közbeni véletlenszerű megváltoztatása, a „zajkeltés”.

A sabotázst algoritmikus védelemmel nem tudjuk megakadályozni, azonban felfedhetjük. Ennek módszerei az **üzenet-** és **párbeszéd-hitelesítés**, amellyel az adatfolyam sorrendisége és időbelisége ellenőrizhető. Mivel az adatok megismerése nélkül a behatoló ezeket az ellenőrzéseket nem tudja kijátszani, a sabotázskísérlet nem marad észrevétlenül.

A behatoló megkísérelheti, hogy az adatkapcsolatba beékelődve az egyik vagy mindkét kapcsolatban álló fél partnerének helyébe lépjen. A sikeres **megszemélyesítés** során a behatoló olyan adatokhoz juthat, amelyeket később már „szabályosan” is felhasználhat célja elérésére. A megszemélyesítés veszélye a kapcsolat felvételekor (illetve felvétele közben) a legnagyobb, éppen ezért külön módszercsoport, a **partnerazonosítás** hivatott megakadályozni a kapcsolatba lépni szándékozó partnerek megszemélyesítését. (Figyelem! A felhasználóazonosítás tulajdonképpen egyfajta partnerazonosítás, az utóbbi azonban szélesebb hatáskörű, a számítógépes rendszer minden aktív elemét — szoftver és hardver algoritmusokat — egyaránt szolgálja.)

Rejtjelezéssel a mindennapi életben oly fontos szerepet betöltő aláírás megfelelőjét, a digitális kézjegyet is előállíthatjuk. A **digitális kézjegy** olyan leképezés, amelynek eredménye — az „aláírás” — mind a leképezés végrehajtójára — az „aláíróra” —, mind a leképezéssel érintett adatra — az „aláírt szövegre” — egyértelműen jellemző.

Az aláírás, az aláírt szöveg és az aláíró kilétének összetartozása egyszerűen és az aláíráskor természetesen formált és titkosan kezelt kulcs ismerete nélkül megállapítható. A digitális kézjegy nem is hamisítható, vagyis még számos

(aláírt szöveg, aláírás)

pár ismeretében sem képes egy kívülálló más felhasználó aláírását előállítani. A digitális kézjegy tehát megfelelő rendszertechnikai környezetben a szerzőség és felelősség rögzítésének, megállapításának fontos eleme lehet.

A rejtjelezés különböző témaköreivel és alkalmazásaival a 2–7. fejezetekben foglalkozunk.

Hozzáférés-védelem

A számítástechnikai berendezések a védelem szempontjából az adatközlő hálózattal szemben a másik végletet jelentik: fizikailag és/vagy ügyvitelileg jól védett elemek, amelyek belsőjében a tárolt és feldolgozott adatok védelméhez nincs szükség rejtjelezésre. A védelmet rejtjelezés helyett a hozzáférési utak ellenőrzésével és felügyeletével valósíthatjuk meg. Erre az ad lehetőséget, hogy az egyes felhasználók igényeit a számítógépes rendszerben futó programok, folyamatok, elégítik ki. Ezek viszont teljesen a rendszer felügyelete alatt állnak, tevékenységüket a rendszer kötelező érvénnyel képes szabályozni.

A hozzáférés-védelmet a 8. és 9. fejezet tárgyalja.

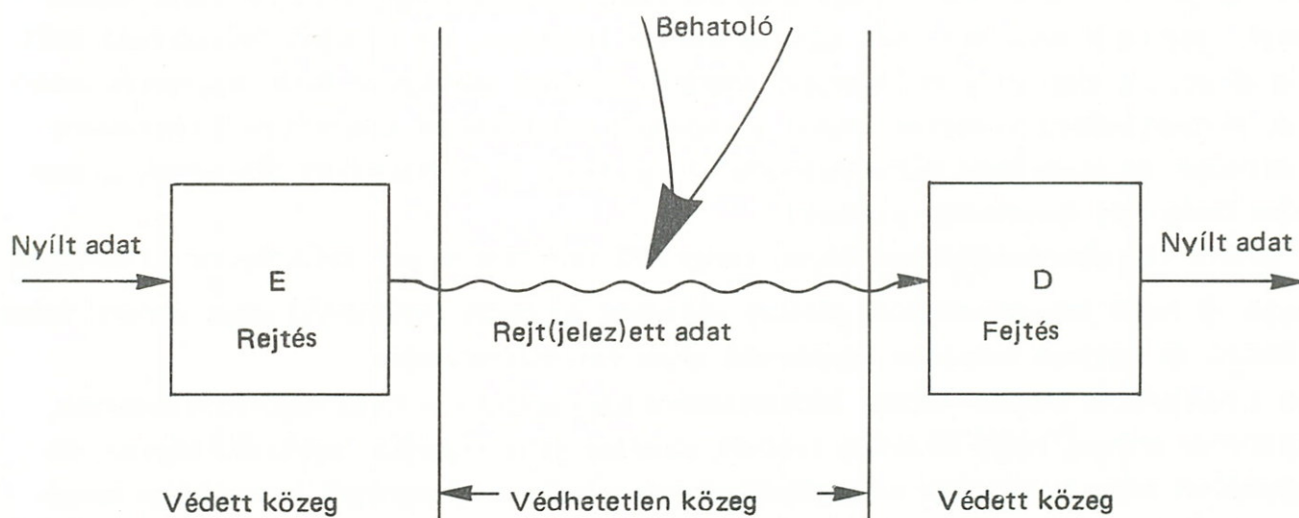
2. Rejtjelezés

2.1 Alapfogalmak

Számítógépes rendszerekben több olyan adattároló és mozgató közeget is találunk, amelyek fizikai védelmét hatásosan nem tudjuk megoldani. Ilyen közeg elsősorban az adatközlő (hírközlő) hálózat, amelynek védelmét térbeli kiterjedése teszi lehetetlenné, de ugyanide sorolhatók azok az adathordozók is, amelyeket a számítógépes konfiguráció védett környezetéből (pl. gépteremből) hosszabb időre eltávolíthatunk. A mágnesszalag és a mindinkább elterjedő hajlékony mágneslemez ez utóbbi típus jellegzetes képviselői. Ezeknek az adathordozóknak a használatát adminisztratív eszközökkel nem lehet egykönnyen szabályozni, de még csak nyomon követni sem! Fizikailag védtelen közegben az adatok védelmét csak egyféleképpen oldhatjuk meg: a védtelen közegbe helyezés előtt úgy transzformáljuk őket, hogy eredeti formájuk a visszaállításra szolgáló transzformáció teljes ismerete nélkül meghatározhatatlan legyen. A védelemnek ezt a formáját **rejtjelezésnek** nevezzük.

A rejtjelezés tehát (2.1 ábra) egy **rejtésből** (E) és **fejtésből** (D) álló transzformációpár (E, D) alkalmazását jelenti. A védhetetlennek minősülő közegre a továbbítandó, tárolandó **nyílt** (formájú) X adat helyett annak **rejtett** (**rejtjelezett**) alakját, $Y=E(X)$ -et bízuk. Ebből a felhasználás előtt az eredeti adatformát a fejtés alkalmazásával, $X=D(Y)$ állítjuk elő. A teljes rendszerrel kapcsolatban **alapvető** feltételezésünk, hogy a behatoló nem ismeri a fejtő transzformáció minden meghatározó adatát, és a titokban tartott részletekre a számára könnyen (vagy egyáltalán) hozzáférhető információkból következtetni nem, vagy csak gazdaságtalanul képes. Az előbbi esetben a rejtjelező rendszer **feltétlen**, egyébként csak **gyakorlati védettség**et nyújt.

A **gyakorlati védettség** fogalmát azonban tovább kell finomítanunk. Ha ugyanis a



2.1 ábra.

védettség nem feltétlen, a behatoló az illetéktelenül megszerzett rejtjelezett adatokat és egyéb „háttér”-információkat felhasználva kellő mennyiségű számítókapacitás birtokában „előbb-utóbb” — vagyis véges idő alatt — célhoz ér. Abszolút értelemben tehát a rendszer nem ad védettséget! A védelemre szoruló adatok értékét azonban korlátozza a birtoklásukkal elérhető haszon, és folyamatosan csökkenti az elkerülhetetlenül bekövetkező elavulásuk. A tözsdei információk értéke például igen nagy lehet, élettartamuk azonban sokszor csak percekben mérhető, de legfeljebb néhány óra. Egyes közigazgatási szintű adatok viszont évekig illetékességérzékenyek maradnak.

A gyakorlati védettség biztosítása tehát azt jelenti, hogy a behatolót a megszerzendő adatok értékét, illetve élettartamát meghaladó mértékű erőforrás- vagy időigényű feladat elé állítjuk. A gyakorlati védettség gondolata két szempontból is jelentős:

- a feltétlen védettséget nyújtó egyetlen rejtjelezési módszer (egy alkalommal használt kulcsfolyam) számítógépes környezetben alkalmazhatatlanul nehézkes,
- a gyakorlati védettség szintjének növekedésével együtt növekszik a rejtjelezés használatának költsége is. A gyakorlati védettség tehát lehetővé teszi, hogy a védelem alkalmazásának költségeit a védendő adatok értékéhez és elavulási sebességéhez igazítsuk.

A rejtjelezés feladatának megfelelően a rejtés és a fejtés egyaránt igen összetett transzformáció. Hatékonyan tehát csak speciális berendezésekkel valósíthatók meg. Ez azonban már önmagában is kizárja, hogy a védelemhez a teljes D (és ha kell az E) transzformáció titkossága szükséges legyen.

Egy védelmi rendszer hatásossága ugyanis csak olyan információ titkosságán múlhat, amely könnyen lecserélhető, ha illetéktelenek számára is ismertté válik. A rejtjelező berendezések pedig csak nehezen (gazdaságtalanul) cserélhetők, és védelmük is problematikus. Sok különböző rejtjelező berendezés gazdaságos előállítását megoldhatatlan feladat, azonos berendezések alkalmazása viszont a behatolási alkalmak számát, illetve az eltulajdonítás sikerének valószínűségét növeli. A rejtjelező berendezések egyébként is a szükségesnél hosszabb ideig vannak kitéve a behatolásnak, mert működési szüneteikben rendszerbeli helyükről nem mozdíthatók el.

Számítógépes rendszerekben az előbbiekhöz még egy további érv járul. Egy rendszerben egyidejűleg több ezer rejtjelezett kapcsolat él (beleértve az adattárolás-visszanyerés folyamatát is), amelyeknek szükségszerűen egymástól is védetteknek kell lenniük. Az E és D transzformációkat tehát egy potenciálisan hatalmas halmazból kell választanunk. Ez azonban csak úgy lehetséges, ha a felhasználók hasonló, és csupán néhány paraméterben eltérő rejtjelező transzformációkat alkalmaznak. Az ilyen rejtjelező rendszer közös paraméterei, jellemzői azonban csak úgy tarthatók titokban, ha minden felhasználó elől eltitkoljuk őket. Ez viszont a felhasználó részéről bizalmi válsághoz és a rejtjelezés alkalmazásának elutasításához vezethet, mivel a védelem mértékének személyes értékelésére nincs lehetősége, és arról sem bizonyosodhat meg, hogy a rendszerben nincsenek „csapdák” (előre beépített behatolási pontok).

A fentiek figyelembevételével olyan rejtjelező rendszerre van szükségünk, amelyben

- a rejtő és fejtő transzformáció jórészt rögzített és nyílt (mindenki által ismert) lehet
- a változó és egyben titkosan kezelendő algoritmuskomponens
 - csak a rejtjelezés (rejtés—fejtés) időtartamára kapcsolódik a nyílt algoritmusvázhhoz,
 - elég tömör ahhoz, hogy kezelése (védett tárolása és mozgatása) egyszerű legyen; de
 - elegendően nagy halmazból választható, hogy a halmaz elemeinek kimerítéses sorra-vételezését alkalmazó megfejtési kísérlet ne lehessen célravezető, illetve a számító-

gépes rendszer üzemeltetéséhez elegendő, egymástól is védett kapcsolatot lehet létrehozni.

Ezeket a követelményeket egy olyan $E(k,x)$, $D(k,x)$ rejtjelező rendszerrel valósítjuk meg, amelynek változtathatóságát a k paraméter értéktartományából kiválasztott (K', K'') rejtő-fejtő kulcspár, titkosságát pedig egyedül a fejtő K'' kulcs titkossága biztosítja (2.2 ábra). Egy nyílt adat rejtése tehát $Y = E(K', X)$, a fejtés pedig $X = D(K'', Y)$. Más képpen megfogalmazva, az $E(\cdot, \cdot)$ és $D(\cdot, \cdot)$ függvényhalmazokból a (K', K'') kulcspár kiválaszt egy összetartozó $E_{K'}(\cdot) = E(K', \cdot)$ és $D_{K''}(\cdot) = D(K'', \cdot)$ transzformációpárt, amelyekkel a rejtés $Y = E_{K'}(X)$, a fejtés $X = D_{K''}(Y)$ módon történik.

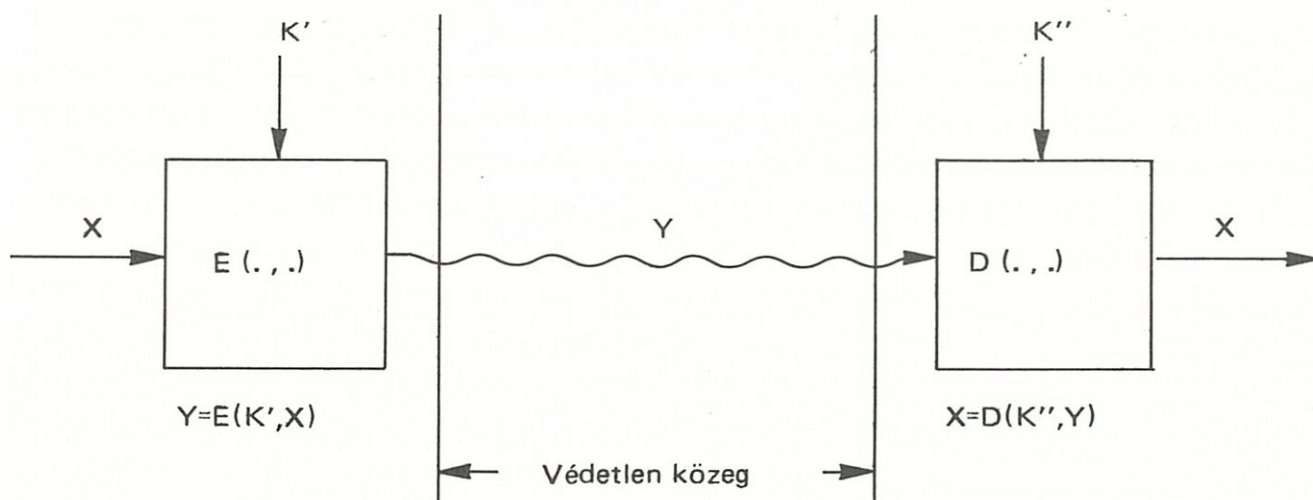
A gyakorlatban rejtés előtt a nyílt szöveget kezelhető és azonos méretű egységekre, **blokkokra** bontjuk. (Szükség esetén az utolsó egységet a blokkméret elérése végett „kipárnázzuk”.) A rejtő transzformáció a feladatát ezeken a blokkokon mint oszthatatlan egységeken hajtja végre. Minden **nyílt blokk** feldolgozásának egy **rejt(jelez)ett blokk** az eredménye. Ha az egyes nyílt blokkok rejtése egymástól független (minden rejtett blokk egyetlen nyílt bloktól és a kulcstól függ), akkor az eljárást **blokkrejtésnek** nevezzük. Ha az egymásra következő rejtett (és nyílt) blokkok között kapcsolat van — tehát egy rejtjelezett blokk önmagában nem fejthető meg —, akkor **folyamatos (blokk) rejtésről** beszélünk. Mint látni fogjuk, bármilyen blokkrejtő eljárásból egyszerű módszerekkel kialakíthatunk folyamatos rejtést.

A rejtjelező rendszereket a rejtő- és fejtőkulcsok viszonya alapján két csoportba soroljuk.

a) **Konvencionális eljárások.** A fejtőkulcs a rejtőkulcs ismeretében különösebb nehézség nélkül meghatározható. A védelem érdekében tehát mindkét kulcsot titkosan kell kezelnünk. A gyakorlatban ebből a csoportból olyan eljárásokat használunk, amelyekben a két kulcs azonos.

b) **Nyilvános (rejtő) kulcsú eljárások.** A két kulcs különböző, és a rejtőkulcsból a rejtő — $E(\cdot, \cdot)$ —, valamint a fejtő — $D(\cdot, \cdot)$ — transzformációk teljes ismeretének birtokában sem lehet a fejtőkulcsra másképpen következtetni, mint ez utóbbi értékkészletének próbálgatásos kimerítésével. A nyilvános kulcsú rendszerekben, mint látni fogjuk, a rejtjelezett kapcsolatok létrehozását egyszerűsíti a fejtőkulcsnak a rejtőkulcstól való „leárnýékoltsága”.

A fejtőkulcs hozzáférhetetlensége semmilyen formában és mértékben nem függ a rejtő és fejtő transzformációk — $E(\cdot, \cdot)$ és $D(\cdot, \cdot)$ — ismeretének mélységétől, részletességétől. Mindkettő teljes egészében, a tervezésükkor és vizsgálatukkor felhasznált ösz-



2.2 ábra.

szes megfontolással együtt, a behatoló rendelkezésére állhat. Ez a tény lehetőséget ad a behatolónak arra, hogy a megfigyelt rejtjelezett kapcsolatból kinyert adatok, információk segítségével algoritmikus támadást indítson a kapcsolat titkos fejtőkulcsának megszerzésére. Ezt a műveletet **rejtjelfejtésnek** nevezzük. A rejtjelezési eljárásokat aszerint szoktuk minősíteni, hogy milyen adatok birtokában indított rejtjelfejtési kísérletnek képesek ellenállni. A rejtő és fejtő transzformációkon kívül a behatolónak a rejtjelfejtéshez általában még további háttér-információi is vannak. Ismerheti a nyílt szöveg nyelvét, például betűinek és szótagjainak gyakoriságát, az információcsere témakörét, és ebből a valószínűleg előforduló szavakra, kifejezésekre következtethet. Építhet a nyílt adatok redundanciájára: élő nyelvi szöveg csak értelmes lehet, az ASCII ábrázolású alfanumerikus karakterek paritásának párosnak kell lennie, a programszövegek meglehetősen kötött szerkezetűek, a rejtjelkulcsok a megjegyezhetőség végett gyakran alfanumerikus szövegformájúak stb. Ezeket az információkat nem „tagadhatjuk meg” a behatolótól, mert titkos kezelésük a teljes rendszer működését a használhatatlanságig merevvé tenné. A háttér-információk tehát a rejtjelezési rendszerek osztályozási alapjaként nem jöhetnek szóba. A rejtjelfejtési módszereket magából a rejtjelezett kapcsolatból rendelkezésre álló információk minősége szerint sorolhatjuk négy osztályba.

A behatoló helyzete akkor a legelőnytelenebb, ha csak az elfogott rejtjelezett adatok állnak a rendelkezésére. Ilyenkor a rejtjelfejtésben csak a háttér-információk felhasználásával remélheti a titkos kulcs megszerzéséhez vezető út lerövidítését. Ezek azonban csak valószínűségi alapon, a rejtjelfejtő „megérzéseire” is támaszkodva érvényesíthetők. A **csak rejtjelezett adatokból** induló rejtjelfejtési kísérlet tehát meglehetősen gyenge támadást jelent, így sikere a szóban forgó rejtjelező eljárást teljesen „lejárhatja”.

Lényegesen előnyösebb helyzetben van az a behatoló, aki a rejtjelezett kapcsolatból **összetartozó nyílt és rejtett adatként** birtokába jut. Ezzel ugyanis a fejtő transzformáció hatását számos pontban ismeri, és ebből kiindulva kísérlelheti meg a teljes transzformáció visszaállítását. A módszert **ismert nyílt szövegű rejtjelfejtésnek** nevezzük, és számítógépes rendszerekben nem is ritkán alkalmazhatjuk. Ennek egyik oka, hogy a számítógépes rendszerben a programok, adatbázisrekordok stb. formájában tárolt információ **igen merev szerkezetű**, ezért a behatolónak már eleve részletes ismeretei vannak a nyílt adatokról (pl. programokban gyakran és meghatározott helyzetben előforduló kulcsszavak stb.). Egy másik, még lényegesebb ok az adatok folytonos elavulása, aminek következtében egy részük nyílt formában is napvilágra kerülhet. Annál is inkább, mert számos rejtjelezett információ a végső rendeltetése szerint valamilyen felhasználói kör számára nyílt formában hozzáférhető. Ha egy rejtjelező eljárás az ismert nyílt szövegű rejtjelfejtési kísérletnek nem képes ellenállni, akkor közzététel előtt az információ megjelenési formáját át kell alakítani. (Egy szöveget például át kell fogalmazni.) Mivel ez számítógépes rendszerekben automatikusan és kötött (tehát kiismerhető!) szabályok mellőzésével nehezen valósítható meg, céljainknak egyetlen rendszer sem felel meg, amely nyílt szövegek ismeretére érzékeny.

A behatoló még előnyösebb helyzetben van akkor, ha maga választhatja meg azokat a nyílt szövegeket, amelyek rejtjelezett megfelelőjét látni kívánja. Ez a **választható nyílt szöveg** alkalmazása rejtjelfejtésben és számítógépes rendszerekben szintén elképzelhető, ha a behatolónak sikerül a rendszer egyéb (nem rejtjelezési) védelmi vonalai mögé kerülnie. Bár ez a rejtjelfejtési módszer a passzív nyílt szövegismeretet hasznosítónál lényegesen kisebb valószínűséggel alkalmazható, a „bizalomra számot tartó” rendszerek el-

lene is védettséggel kell, hogy rendelkezzenek. (Különben egyebek között fennállhat a rendszer üzemeltetői részéről szervezett visszaélés lehetősége.)

Végül előfordulhat az is, hogy a behatoló bármelyik — akár a nyílt, akár a rejtett — szöveget meghatározhatja, amelynek párját látni kívánja. A rejtjelező eljárással szemben ez a támadás elvben még szigorúbb követelményeket támaszt. A gyakorlatban azonban a választható nyílt szövegű rejtjelfejtésnek ellenálló rejtjelező eljárások ezzel a választható szöveg birtokában indított támadással szemben is védelmet nyújtanak.

2.2 Rendszerteknikai problémák

A rejtjelezési eljárás — még ha alapvető is — a védett kapcsolatok üzemeltetésének csupán egy eleme. Önmagában elsősorban a passzív behatolás, a lehallgatás ellen véd. Egy behatoló azonban aktívan is beavatkozhat a kapcsolat működésébe. Célja szerint az aktív behatolás lehet **szabotázs** — a behatoló célja zavarkeltés, a kapcsolat működésének akadályozása —, vagy lehet **megtévesztés** — kísérlet arra, hogy a rejtjelezett kapcsolatba lépni kívánó vagy kapcsolatban álló partnerek valamelyikét védendő információi, adatai kiadására rávegye. A szabotázs elkövetésével a behatoló arra törekszik, hogy a kapcsolaton áramló adatokat lehetőség szerint a kapcsolatban álló partnerek számára felfedezhetetlen módon megváltoztassa. A szabotázs eszközei tehát:

- rejtjelezett üzenetek kivonása a kapcsolatból és
- korábban kivont rejtjelezett üzenetek beiktatása az üzenetfolyamba egy későbbi időpontban.

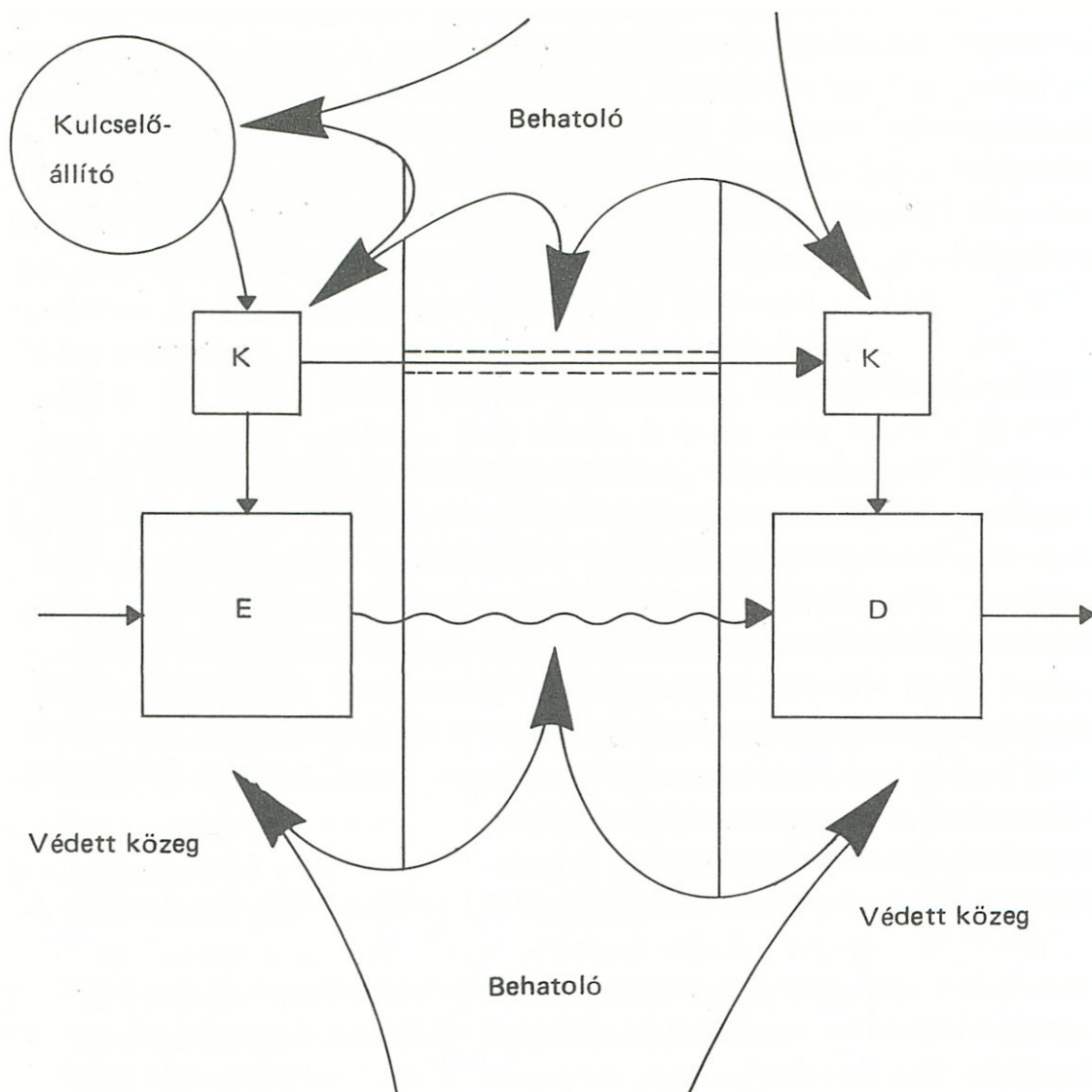
A megtévesztés minden formájának a **megszemélyesítés** a módszere. A behatoló megkísérli elhitetni a szemben álló féllel (számítógép, felhasználó), hogy a szándékoltnak partnere. Sikeres megszemélyesítéssel bizalmas adatokhoz, esetleg az egész kapcsolat védettségét megszüntető információkhoz (pl. a kapcsolat kulcsa) juthat. A megtévesztés egy számítógépes rendszer működésének számos szakaszában és sokféle formában kísérelhető meg. Az egyik formája azonban módszerében nagyon közel áll a szabotázshoz. A behatoló lehallgatja és elraktározza egy-egy adott előfizetőnek a bejelentkezéskor a rendszerrel folytatott párbeszédét. Egy későbbi időpontban — anélkül, hogy értelmét megfejtette volna — a párbeszéd felhasználótól eredő részét megkísérelheti bejelentkezésre felhasználni. Ha a megtévesztés sikerül, akkor a behatoló az utánzott felhasználóként jelenik meg a rendszerben, és így annak bizalmas információit eltulajdoníthatja.

A szabotázs felismeréséhez és a megtévesztés leleplezéséhez tehát egyaránt szükséges a rejtjelezett kapcsolat időbeli folytonosságának figyelemmel kísérése. Egy rejtjelező alapeljárás önmagában erre lehetőséget nem teremt, mert

- az üzenetek rejtése (fejtése) egymástól függetlenül történik; és
- a (szinte egyeduralkodóan elterjedt) blokkrejtő rejtjelezés még egyetlen üzenetet is általában több független blokként kezel.

Az időbeli folytonosság pedig csak akkor valósítható meg, ha az üzenetek sorrendi folytonossága a kapcsolat bármelyik irányában és a teljes párbeszédben is ellenőrizhető. A megtévesztés lehetőségének kizárása végett a rejtjelezett kapcsolat felvétele közben ezenkívül még a partner azonosságának hitelt érdemlő megállapíthatóságára is szükség van. A rejtjelezési eljárással tehát ötvöznünk kell a **kapcsolathitelesítés** módszereit is.

Összetett rendszerteknikai problémák forrása a védett kapcsolat kulcsának kezelé-



2.3 ábra.

se is. A 2.3 ábrán feltüntetett módon, a kapcsolat kulcsa élete folyamán többféle műveletnek is tárgya. Ezeket együttesen **kulcsgondozásnak** nevezzük. Egy kulcs élete az előállításával kezdődik. Az előállított kulcsot a kapcsolat mindkét végpontjához (mindkét partnerhez) el kell juttatnunk. A kulcsot alkalmazása közben a rejtjelező berendezésben, használaton kívül pedig a rendszer egyéb helyén tároljuk. Eközben védelméről annak tudatában kell gondoskodnunk, hogy a **kapcsolat véletlenszerűsége semmiképpen sem haladja meg kulcsa véletlenszerűségének mértékét!** A kulcsgondozó rendszer ugyanakkor természetes és elsődleges támadási célpont, mert sokféle behatólási alkalmat és módot kínál.

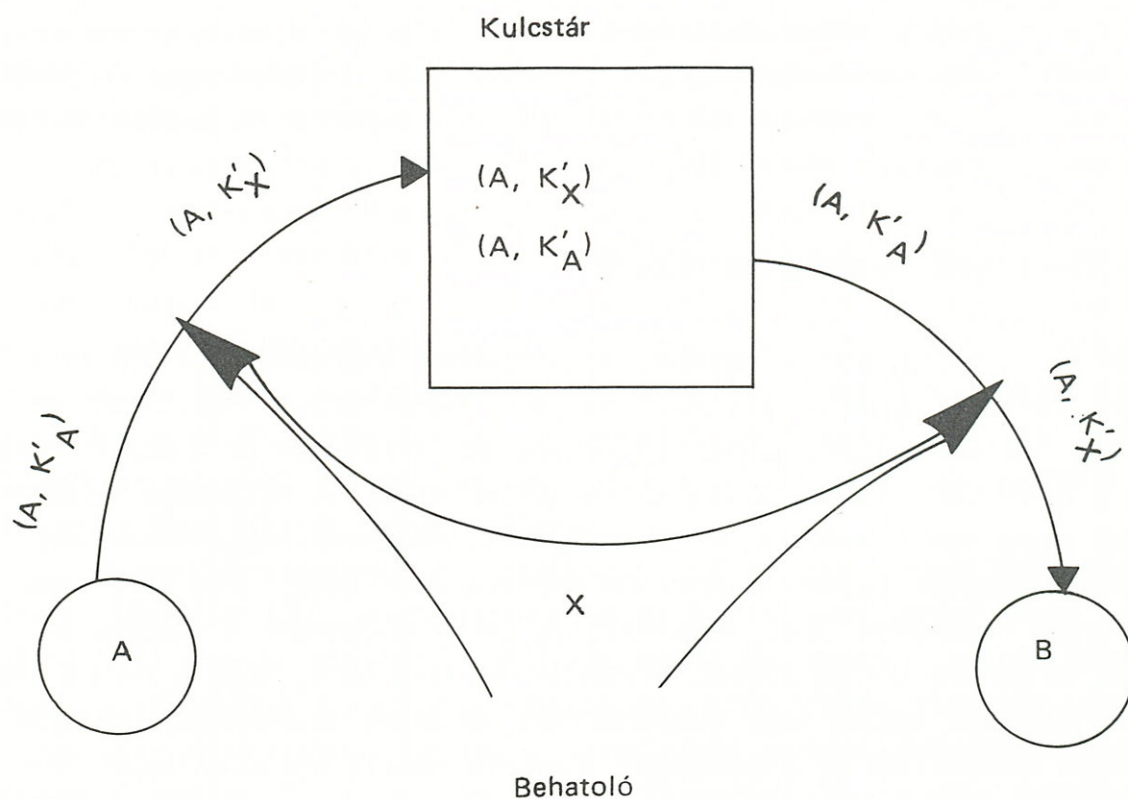
A legnagyobb hasznot természetesen a **kulcselőállítás** folyamatának birtokbavételével érheti el a beható, mert ezzel minden kapcsolat kulcsát közvetlenül kézben tarthatja. A kulcselőállítás folyamatának védelmét tehát minden eszközzel (fizikai, ügyviteli, algoritmikus) biztosítani kell. A kulcs véletlenszerűségét a közvetlen behatólás mellett a kulcselőállítás algoritmus „kiismerhetősége” is veszélyezteti. Jogosan feltételezhetjük ugyanis, hogy a beható a kulcselőállítás algoritmus vázát ismeri, és saját használatára jogosan megszerezhet számos egymást követően előállított kulcsot. Ha ezeknek az adatoknak a birtokában a sorban következőnek előállított kulcsokra következtetni tud, akkor a kulcselőállítás minden egyéb védelmi intézkedés ellenére védtelenné vált!

A legnehezebb rendszertechnikai feladatot azonban a kulcsok célba juttatása, a **kulcskiosztás** jelenti. Nyilvánvaló ugyanis, hogy a kulcsokat védett csatornákon kell moz-

gatnunk, és az adatkapcsolat védeltségének mértéke soha sem haladja meg a kulcsát továbbító csatornáét. (Mert a behatoló mindig a kevésbé védett csatornába kísérel meg be- törni!) Az ellentmondást az kelti, hogy számítógépes rendszerekben a kulcsok továbbítá- sára ugyanaz a (védetlennek feltételezett) közeg áll csupán rendelkezésre, mint az egyéb kapcsolatok létrehozására. Elképzelhető ugyan a számítógépes rendszeren kívül kulcsköz- vetítő rendszer alkalmazása (pl. postai küldemény, személyes kulcsfelvétel stb.), ezek a- zonban a kulcsmozgatás sebességét jelentősen lecsökkentik, és ezért az új kulcsok hasz- nálatbavételét elfogadhatatlan mértékben korlátozzák. A végső eredmény mindig az len- ne, hogy a rendszer egy használójának minden lehetséges partneréhez hosszú ideig érvé- nyes külön kulcsot kellene tárolnia. A felhasználónkénti nagy mennyiségű kulcs védett kezelése azonban megoldhatatlan feladatot jelent, új felhasználó felvétele vagy régi fel- használók kizárása pedig elviselhetetlenül nehézkessé válik.

A konvencionális (azonos kulcsú) rejtjelező eljárást hasznosító rendszerekben a kulcsszétosztást csak ún. **alapkulcsrendszerre** támaszkodva lehet védetten megszervezni. A számítógép telepítésével egyidejűleg kizárólag a kulcsok továbbítására szolgáló rejtjelezett kapcsolatokat is kiépítünk, amelyek kulcsai az **alapkulcsok**. Az alapkulcsokat a **rendsze- ren kívül és védett körülmények között juttatjuk** felhasználási helyükre, és élettartamuk alatt a berendezések **különlegesen védett részében tároljuk** őket. Az alapkulcsrendszert a védeltség fokozása érdekében **időről időre módosítjuk**. Minden esetben a rendszeren kí- vüli kulcsváltoztatást alkalmazzuk, ez azonban a módosítások kis gyakorisága miatt most nem okoz jelentős nehézséget.

A nyilvános rejtőkulcsú eljárások látszólag mentesek a kulcsszétosztás fenti problé- máitól. Egy felhasználó saját rejtő-fejtő kulcspárját védett körülmények között állíthatja elő, és ezek közül csak az egyébként is mindenki számára hozzáférhetővé teendő rejtő- kulcsot kell a partnerhez – vagy egy központi helyre, ahol mindenki elérheti – eljuttat- nia. Amint azonban arra a figyelmünket a 2.4 ábra felhívja, a védetlen közegben a behatoló a továbbított rejtőkulcsot akár a központi tárolóhely felé, akár onnan a kap-



2.4 ábra.

csolatot kereső partner felé továbbítás során a saját rejtőkulcsára cserélheti ki. Ez azonban azt is jelenti, hogy egyetlen felhasználó sem lehet biztos partnere kilétében, a rendszer **továbbra is védtelen!**

A rejtjelezés alkalmazásának az a következménye, hogy egy-egy továbbított üzenetet csak az alkalmazott kulcs birtokosa(i) állíthat(nak) elő, azt sugallja, hogy a rejtjelezés célirányos alkalmazásával megteremthető a hagyományos rendszerekben okmányt hitelesítő aláírás számítógépes megfelelője is. A **digitális kézjegy**, mint hagyományos megfelelője az aláírás, a következő tulajdonságokkal kell, hogy rendelkezzen:

- egyértelműen jellemző mind az aláírt üzenetre, mind az aláírójára;
- **csak** az üzenet kezdeményezője (tulajdonosa) képes előállítani;
- hitelességét a rá vonatkozó külső információ alapján a címzetten kívül a rendszertől függően bármely harmadik személy vagy az arra képesített harmadik személyek valamelyike egyszerűen ellenőrizheti.

A digitális kézjegy tehát alkalmas a számítógépes rendszer felhasználói közötti adatkapcsolatban előálló vitás helyzetek rendezésére. A digitális kézjegy előállításának módszereivel a későbbiekben foglalkozunk. Itt még arra kell felhívni a figyelmet, hogy a digitális kézjegy – bármilyen tökéletes is legyen – **nem azonos értékű** a hagyományos aláírással. Egy digitális kézjegy „tulajdonosa” ugyanis hivatkozhat arra, hogy elvesztette vagy eltulajdonították tőle. A valódi aláírással ez természetesen nem történhet meg!

Az eddigiekben a rejtjelezés alkalmazásának technikai jellegű kérdéseit említettük. Ezekről elkülönülő és fontos témakör a rejtjelezés számítógépes rendszerbe építésének stratégiája. Az egyik végletet az jelenti, amikor a **rejtjelezés elkülöníthetetlen része** a rendszernek, és a felhasználók számára csak a rendszer megnövekedett biztonságán keresztül érzékelhető. Egyébként a rejtjelezés teljes átlátszó, alkalmazása a rendszerben automatikus és kikerülhetetlen. A másik végletet a **rejtjelezést egyáltalán nem támogató** rendszerek képviselik. A rejtjelezés a felhasználó tulajdonában levő berendezésben történik, a közösen használt rendszer legfeljebb a védett tárolás és feldolgozás lehetőségét nyújtja. Szélsőséges esetben azonban csak a hozzáférés-védelem nélküli tárolás és a felhasználók közötti védtelen kapcsolat közvetítő eleme. Közbenső helyet foglalnak el azok a rendszerek, amelyek **támogatják a rejtjelezés alkalmazását**, de használatát a felhasználó döntésére bízzák. Az automatizmus és kényelem tekintetében ezek a rendszerek is jelentős mértékben különbözhetnek egymástól.

2.3 Néhány hagyományos rejtjelezési eljárás

A rejtjelezés minden valószínűség szerint egykorú az emberiség írásbeliségével. Ha ez a feltevés bizonyíthatatlan marad is, a rejtjelezés alkalmazásáról meglehetősen régi, hitelt érdemlő híradás tudósít: már Julius Caesar használt rejtjelezést, amelyben az ábécé minden betűjét a (ciklikusan) következő harmadik betűvel helyettesítették. A rejtjelezés módszerei az idők folyamán egyre tökéletesedtek, a számítógépes rendszerek napjainkban végbemenő általános térhódításának kezdetéig azonban szinte kivétel nélkül csak alfanumerikus szövegek védelmére használták őket, alkalmazásuk pedig elsősorban a katonai és diplomáciai hírközlés területére korlátozódott. Mindezek és az a tény, hogy a rejtést maguk a nyílt üzenetek előállítói kézzel vagy meglehetősen nehézkes mechanikus szerkezetekkel végezték, erősen korlátozták az alkalmazott transzformációk bonyolultságát. Ma már ezek a módszerek nem is képesek ellenállni a nagy sebességű számítógépekkel támo-

gatott rejtjelfejtési kísérleteknek. Közülük néhányat röviden mégis bemutatunk, mert a bennük felhasznált alapötleteket a jelenlegi hatékonyabb rejtjelezési eljárások építőelemként felhasználják, és gyengeségeik is tanulságosak.

Helyettesítés

A legegyszerűbb és legkézenfekvőbb rejtjelezési eljárás a nyílt szöveg minden egyes betűjének egy másik betűvel való **helyettesítése**. A megfejthetőség végett két különböző nyílt szövegbeli betűt nem helyettesítünk ugyanazzal a betűvel. A rejtjelezés kulcsa tehát az alkalmazott **ábécé egy permutációja**. Az ábécé betűit eredeti és a permutációnak megfelelő sorrendben egymás alá írva, az egymás alá kerülő betűk helyettesítő párokat alkotnak. Például az

AÁBCDEÉFGHIJKLMNOÖPQRSTUÜVWXYZ
BWEKÉQFMVYÖALUCONÁÜPHISDXTRGZJ

permutáció alkalmazásakor az ELREJTÉS rejtett megfelelője a QUHQASFI, míg a CQVMQASFI megfejtése a MEGFEJTÉS. A megfejtés megkönnyítésére felírhatjuk az előbbi permutáció inverzét (az alsó sor rendezésével):

JÖAMUBDÉXRSZCKFONIQEWTVLPGÁÜYH
AÁBCDEÉFGHIJKLMNOÖPQRSTUÜVWXYZ

Egy pillanatig úgy tűnhet, hogy a helyettesítés jó védelmet ad, hiszen egy 30 betűs ábécének $30! \sim 3 \cdot 10^{32}$ különböző permutációja van. A kulcstér kimerítése tehát reménytelennek látszik. Ezt a helyettesítési típust mégis szinte gyerekjáték megfejteni! A helyettesítés ugyanis nem tünteti el az eredeti szöveg nyelvének statisztikai és szerkezeti tulajdonságait: a leggyakoribb nyílt szövegbeli betű megfelelője a leggyakoribb a rejtett szövegben; a ragoknak, kötőszavaknak, kettősbetűknek megfelelő betűkombinációk a rejtett szövegben is ismétlődő betűcsoportokat alkotnak; egy-egy szón belül ismétlődően előforduló magánhangzók vagy kettőzött mássalhangzók olyan szóvázat képeznek, amely a rejtett szövegben is megjelenik stb.

Az egyszerű helyettesítésnek ezt az előnytelen tulajdonságát sokféleképpen megkísérelték megszüntetni. Ezek közül az egyik módszer több permutált ábécé ciklikus alkalmazása, az ún. **periodikus helyettesítés**. Ha például 5 permutált ábécét használunk, akkor minden ötödik betűt ugyanazzal a permutált ábécével rejtik el. Másképpen: az $5n+i$ sorszámú nyílt szöveg betűrejtése az i -edik permutált ábécé szerint történik. A fenti statisztikai és szerkezeti tulajdonságokra rejtjelfejtéskor most közvetlenül nem támaszkodhatunk. Elsőként a permutációk számát, a **periódust** kell meghatároznunk, csak azután alkalmazhatjuk az egyszerű helyettesítésnél hasznosnak bizonyult módszereket a rejtett szöveg periódus által kijelölt részsorozataira.

A periódusmeghatározás módszerét elsőnek egy porosz tiszt, Fiedrich Kasiski tette közzé 1863-ban [2]. A módszer azon az észrevételen alapul, hogy a nyílt szövegben előforduló azonos betűkombinációk a rejtett szövegben is azonos betűkombinációk lesznek, ha távolságuk a periódus egész számú többszöröse. Ilyenkor ugyanis az ismétlődő betűkombináció egy-egy betűjét a kombináció minden előfordulásában ugyanazzal az ábécével rejtjük. Gyakran ismétlődő betűkombinációk minden nyelvben találhatók, a ma-

gyar nyelvben ezek például a ragok, képzők, kötőszavak stb. Ha például 5 permutált ábécét használunk, a SÁG főnévképző minden előfordulása ugyanazt a betűkombinációt eredményezi, valahányszor szövegbeli távolságuk 5 egész számú többszöröse. Ilyenkor mindegyik előfordulás S betűjét ugyanaz az ábécé rejti, és minden előfordulás Á betűjét is azonos (de az előzőtől különböző!) ábécé rejti stb.

A rejtjelfejtő tehát ismétlődő betűkombinációkat — általában hármasokat, ritkábban párokat, négyeseket — keres a rejtjelezett szövegben. Periódusként azok az egész számok jöhetnek szóba, amelyek az azonos kombinációk közötti távolságok zömének osztói. Ezzel a módszerrel a lehetséges periódusértékek köre jelentősen korlátozható. A periódusjelöltek száma tovább szűkíthető, a következőképpen. Ha a feltételezett periódus n , akkor a rejtett szöveget n részsorozatra osztjuk úgy, hogy az i -edik részsorozat az $n \cdot k + i$ sorszámmú betűket tartalmazza. Minden részsorozatra felvesszük a betű gyakoriságokat, és n -t csak akkor tartjuk meg továbbra is periódusjelöltként, ha minden gyakoriság az egyszerű helyettesítésre jellemző erősen egyenlőtlen elosztást mutatja.

A periódus meghatározásával viszont a rejtjelfejtési feladatot a periódusnak megfelelő számú egyszerű helyettesítés rejtjelfejtési feladatára vezettük vissza. Ez a módszer akkor is sikeres, ha az utolsó menetre még egynél több periódusjelölt marad.

Ha egy nyílt szöveget periodikus helyettesítéssel rejtünk, majd az eredményt egy újabb periodikus helyettesítésnek vetjük alá, az eredmény ismét periodikus rejtés, de általában jóval nagyobb periódusú. Ha a két rejtés I_1 és I_2 periódusai relatív prímek, akkor az eredő helyettesítés periódusa $I_1 \cdot I_2$. A periódusnövekedés következtében a fenti periódusmeghatározáshoz sokkal nagyobb méretű rejtett szöveg szükséges. Bár ezeknek a több periódusú, ún. **Vigenere-rejtjeleknek** a fejtése ezért lényegesen nehezebb feladat, még mindig megoldható [2].

Kulcsfolyamos rejtés

A periodikus helyettesítés megfejtésére periodikussága nyújt lehetőséget. Ennek a gyengeségnek a megszüntetésére a változó helyettesítés szabályát aperiodikussá kell tennünk. Ennek egyik elterjedt megoldása a következő:

A helyettesítést egy olyan szöveg vezérli, amely általában egy könnyen hozzáférhető könyvben található. A szöveg kezdetét a kapcsolatban álló felek által közösen kialakított és függetlenül is alkalmazható algoritmus — ez a **kulcs** — jelöli ki. (A kezdet lehet előre rögzített, vagy függhet a rejtés dátumától stb.) Példánkban a kulcs mutasson a 2.1 fejezet első bekezdésének kezdetére. Rejtjelezéskor a nyílt és vezérlőszöveget betűnként egymás alá írjuk

EZ KULCSFOLYAMOS REJTÉS SZ ÁMITÓGÉPESREN DSZERE

Általános esetben az ábécé minden egyes betűjéhez egy permutált ábécé tartozik. Egy nyílt szövegbeli betű rejtése az alatta álló vezérlőszöveg-betűhöz tartozó permutált ábécé szerint történik. A vezérlőszöveg tehát az egyes helyettesítő ábécék alkalmazási sorrendjét határozza meg, amely így aperiodikus. Esetünkben a helyettesítési szabályt egy 30x30-as mátrix írja le, amelynek minden sora egy permutált ábécé. A helyettesítő betű a nyílt szöveg helyettesítendő betűjéhez tartozó oszlop és a vezérlőszöveg megfelelő betűjéhez tartozó sor közös eleme.

A rejtési művelet egyszerűsítésére az egyes betűkhöz speciális helyettesítési permutációt választunk. Számozzuk az ábécé egyes betűit 1–30-ig. Az i -edik betűhöz permutációként az i lépéses ciklikus eltolást választjuk. Tehát

AÁBCD	EÉFGH	IJKLM	NOÖPQ	RSTUÜ	VWXYZ
5	10	15	20	25	30

és pl. az N betűhöz az

AÁBCDEEÉFGHIJKLMNOÖPQRSTUÜVWXYZ
MNOÖPQRSTUÜVWXYZAÁBCDEEÉFGHIJKL

permutáció tartozik. Mint könnyen belátható, a rejtés a nyílt és vezérlőszöveg betűinek (számértéküknek) mod 30 összegezésével történik. A példa szövege rejtve tehát

XZMGÜWGOUBDTEFWXJYXX

Ugyanis $E+S=6+22=28=X$, $Z+Z=30+30=60=30=Z$ stb.

A megfejtés természetesen a vezérlőszöveg mod 30 kivonásával jön létre.

A kulcsfolyamos rejtés rejtjelfejtésére az 1890-es évek végén Baziers talált megoldást [2], majd 1918-ban Friedman publikált eljárást [2]. A leghatékonyabb módszer azon alapul, hogy a rejtjelfejtő egy vagy több (az üzenet feltételezett témakörétől függő) valószínű szó előfordulására számíthat a nyílt szövegben, és kihasználja azt is, hogy a vezérlőszöveg egyben értelmes szöveg is. Egy valószínű szót a rejtjelező minden lehetséges pozícióban kivon a rejtett szövegből, és figyeli, hogy melyik kivonás eredményez értelmesen kiterjeszthető szövegtöredéket. Ha a kiterjesztés sikerül, akkor az így kapott vezérlőszöveg-részletet kivonja a rejtett szövegből (fázishelyesen!). A feltételezett szónak így egy szöveggörnyezetét kapja meg, amelyet ismét értelmesen kísérel meg kiterjeszteni. Ha ez sikerül, újabb kivonással a vezérlőszöveg már ismert részletének környezete növelhető. Példánkban abból a feltételezésből kiindulva, hogy a nyílt szöveg a rejtjelezésről szól, valószínű szóként a KULCS-ot választjuk. Ezt a harmadik pozícióban a rejtett szövegből levonva, az ÁMITÓ szövegrészletet kapjuk. Ennek (egyik) kiterjesztési lehetősége a SZÁMÍTÓGÉP. Ezt, most már az adott helyen, a rejtjelezett szövegből ismét levonva, nyílt szöveggént az EZKULCSFOL adódik. Logikus kiterjesztése az EZKULCSFOLYAM, amiből a vezérlőszövegre a SZÁMÍTÓGÉPESR adódik. Az utolsó kiterjesztés a SZÁMÍTÓGÉPESRENDSZER, amely a teljes megfejtést — EZKULCSFOLYAM — adja. (Példánk természetesen rövid és mesterkél, célja csupán a bemutatás). Ha a feltételezett szót (esetünkben KULCS-ot) helytelen fázisban vonjuk le, akkor általában kiterjeszthetetlen vezérlőszöveg-részletet kapunk. Példánkban a KULCS-ot a helyes pozíciótól egyetlen hellyel balra levonva, a vezérlőszöveg-részlet WRÜRE-nek adódna, ami nyilvánvalóan képtelenség. Előfordulhat azonban — különösen rövid feltételezett szó esetén — téves találat is. Példánkban a helyes fázistól egyetlen pozícióval jobbra végezve el a kivonást, a kísérleti vezérlőszöveg-részlet VALDÜ, ami a „jóVALDÜhösebb” kiterjesztést megengedi. Ezt újból kivonva azonban az értelmetlen „öxKULCSlnkobd” nyílt szöveghez jutunk, amely a VALDÜ-t eredményező KULCS pozíciót kizárja a további vizsgálatból.

Az ismertetett rejtjelfejtési módszert kombinálhatjuk egy másik (önállóan is alkalmaz-

ható) megközelítési móddal. Példánk rejtett szövegében négyszer fordul elő az X, és ebből háromszor az E–S (illetve S–E) betűpár eredményeként. Ez a megfigyelés a következő megfontoláshoz vezet. Egy rejtjelezett szöveg X betűje 30 betűpárt képviselhet: A–W, Á–V, B–Ü, C–U, D–T, E–S, E–R, F–Q, G–P, H–Ö, I–O, J–N, K–M, L–L, X–Z, Y–Y, illetve ezek **fordított sorrendű megfelelőit**. Ezeknek a betűpároknak az előfordulási valószínűsége – amely az egyedi betűk egymás mellé kerülését független eseményekként kezelve kiszámítható – azonban igen eltérő, és első kísérletnél elegendő csak a gyakori betűkből alkotott párokat figyelembe venni.

Az a tény, hogy a kulcsfolyamos rejtjelezés sikerrel hatástalanítható, azt mutatja, hogy az élő nyelvek legalább 50 % redundanciát tartalmaznak. (Ugyanis egyetlen rejtjelezett szövegből két azonos hosszúságú szövegre tudunk következtetni!) Ez a gondolat a kulcsfolyamos rejtés ismételt alkalmazására is érvényes. Ha a nyílt szöveg 75 %-ban redundáns, a kulcsfolyamos rejtést négyszeres alkalmazása már feltörhetetlenné teszi. Ellenkező esetben ugyanis a rejtjelfejtő egyetlen szövegből ötre (a nyílt szöveg és négy kulcs) tudna következtetni, ami legalább 80 % redundanciára mutatna.

Transzponálás

A nyílt szöveget azonos számú betűt tartalmazó blokkokra bontjuk (az utolsó blokkot a szükséges mértékben kipárnázva), és az egyes blokkokon belül a betűket egy rögzített permutáció szerint átrendezzük. Ötbetűs blokkokat és a (25143) permutációt (első helyre az eredeti blokk második, második helyre az eredeti blokk ötödik stb. betűje kerül) alkalmazva, korábbi példánk rejtett szövege (ETPA kipárnázással) az alábbi lesz:

ZLEUKSLCOFASYOMEÉRTJEASPT

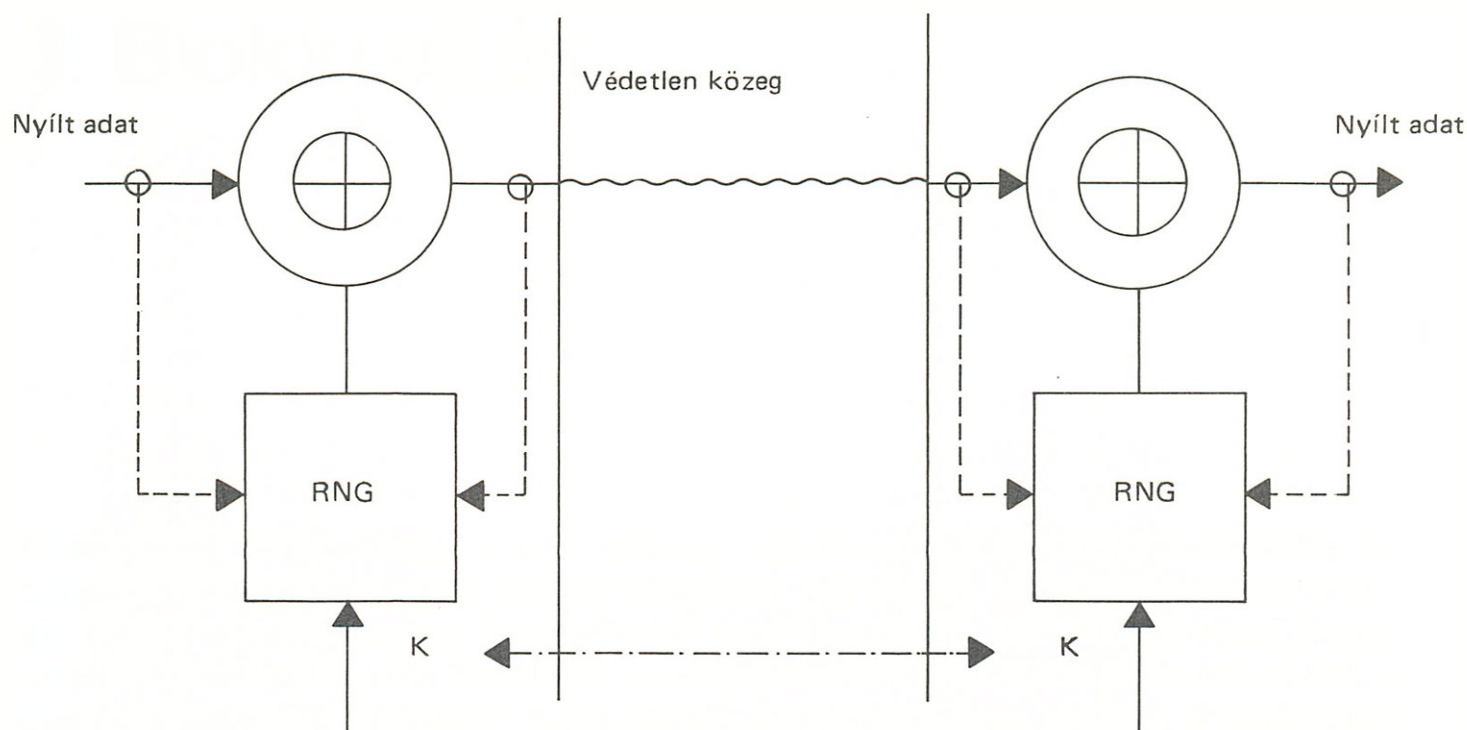
A rejtjelfejtés a betűpárok és -hármak gyakoriságtábláját használja, és olyan permutációk keresését jelenti, amelyek a rejtjelezett szövegre alkalmazva gyakori betűkombinációkat egyesítenek. Ehhez természetesen kísérleti blokkhosszokat kell választanunk. A rejtjelfejtés azonban így sem okoz megoldhatatlan nehézségeket.

Bár a helyettesítés és a transzponálás – mint láttuk – önmagában érdemleges védelmet nem nyújt, fontos alkotóeleme más, jóval hatásosabb rejtjelezési transzformációknak. Ilyenekkel a későbbiekben foglalkozunk.

2.4 Bitrejtés

Egy számítógépes nyílt szöveg legkisebb önállóan (és egymástól függetlenül) rejthető egységei a bitek. A bitenkénti rejtjelezés alapmegoldását a 2.5 ábra szemlélteti. A rejtő és fejtő transzformáció egyaránt a mod-2 összeadás. Az adás helyén a nyílt szöveget képező bitsorozathoz egy másik, a **kulcsfolyamot** képviselő bitsorozatot adunk bitenként mod-2. A vétel helyén **ugyanannak a kulcsfolyamnak ismételt, bitenkénti mod-2** hozzáadásával nyerjük vissza a nyílt szöveget. (A bitenkénti mod-2 összeadás öninverz művelet!)

A kulcsfolyamot egy **véletlenszám-generátorral** állítjuk elő, amelynek indítását a **kulcs** határozza meg. Az adási és vételi oldalon a véletlenszám-generátorok azonosak, in-



2.5 ábra.

dításuk egy időben és azonos kulccsal történik. Mivel a mod-2 összeadás egyszerű transzformáció, a bitrejtés nyílt és rejtett szöveg együttes birtoklása esetén azonnal szolgáltatja a kulcsfolyamot is. (Illetve annak megfelelő szegmensét.) Ezért egy kulcsfolyam **ismételten nem használható**, és ugyanezen megfontolásból **ciklusokat sem tartalmazhat**. Mivel minden algoritmikus véletlenszám-generátor egyetlen indítás után ciklikus sorozatot állít elő, a véletlenszám-generátort időnként **újra kell indítani**, természetesen a két oldalt összehangolva. Ezt a nyílt és (esetleg) a rejtjelezett bitsorozat megfelelő (az ábrán is érzékeltetett) visszacsatolásával érhetjük el.

A bitrejtés egyszerű rejtjelezési módszer, és mivel a kulcsfolyam az átvíendő nyílt szöveggel azonos hosszúságú, alkalmas a kapcsolat időbeli folytonosságának ellenőrzésére is. Számítógépes rendszerekben mégsem alkalmazzuk, elsősorban azért nem, mert igen érzékeny

- az adási és vételi oldal szinkronizmusára és
- a visszacsatolások miatt a vonali hibákra is.

Mindamellett ma már hatékony és „erős” blokkrejtő eljárások állnak rendelkezésünkre.

3. Blokkrejtés

A blokkrejtés alkalmazásakor a nyílt szöveget **azonos hosszúságú blokkokra** bontjuk, és az egyes blokkokat egymástól **teljesen függetlenül** rejtjelezzük. Számítógépes környezetben a nyílt szöveg bitsorozat formájában áll rendelkezésünkre, a blokkok hosszát bitben mérjük. A blokkrejtés lehet **hossztartó** — a rejtjelezett és nyílt blokkok hossza azonos — vagy **redundáns** — a rejtjelezett blokk hossza meghaladja a nyílt blokkét. A blokkméret elrejtéskor **nem csökkenhet**, mert ez nyilvánvalóan információvesztéshez vezetne. A blokkrejtéssel kapcsolatos alapvető felismerés az, hogy minden változata tulajdonképpen egyszerű helyettesítés. Az egyes nyílt B_i blokkokat rögzített szabály szerint a C_i rejtjelezett megfelelőjükkel helyettesítjük. A helyettesítés független a helyettesítés környezetétől, és csak a kulcstól függ. A blokkrejtés tehát (egy adott kulcs mellett) a

$$\begin{array}{ccccccc} B_0 & B_1 & B_2 & & \dots & B_{N-2} & B_{N-1} \\ C_0 & C_1 & C_2 & & \dots & C_{N-2} & C_{N-1} \end{array}$$

táblázattal jellemezhető, a táblázat mérete n -bites blokkméret mellett $N = 2^n - 1$. Hossztartó blokkrejtés esetén a C -k a B -k egy permutációját alkotják (az egyértelmű megfejtettség érdekében), és a már vizsgált egyszerű helyettesítéssel a párhuzam teljes. A blokkrejtés védettségét tehát a blokkméret határozza meg. Kis blokkméret esetén a nyílt szöveg statisztikai és szerkezeti tulajdonságainak ismerete — mint láttuk — már egyedül a rejtjelezett szöveg birtokában is sikeres rejtjelfejtést tesz lehetővé. A blokkméret növelése a blokkrejtés védettségét többféleképpen is növeli. A blokkméret növekedésével a nyilván tartandó statisztikai és szerkezeti adatok mérete exponenciálisan nő; a nyílt blokkok eloszlása az egyenletes felé közelít; és végül számos nyílt-rejtett blokkpár ismerete is csak egy elhanyagolhatóan kis töredékét „leplezi le” a fejtő transzformációnak.

A blokkméret növelésével azonban a helyettesítésre alkalmazható permutációk száma is csökken, mivel a transzformáció táblázatos nyilvántartása lehetetlenné válik. Ezért csak azok a helyettesítési transzformációk jöhetnek számításba, amelyek lényegesen tömörebben leírható formában függenek a kulcstól.

A megoldást a **rejtjelötvözés** szolgáltatja. Ennek alapgondolata, hogy egyszerű — ezért kényelmesen megvalósítható — rejtjelezési transzformációk egymás utáni alkalmazásával kísérelünk meg nagyfokú védettséget előállítani. A gondolat nem új, már az I. világháborúban használt német ADFGVX harctéri rejtjelezés is egy alkalmazását jelenti. Shannon híres, 1949-ben publikált cikkében [2] a rejtjelötvözést egy speciális formában,

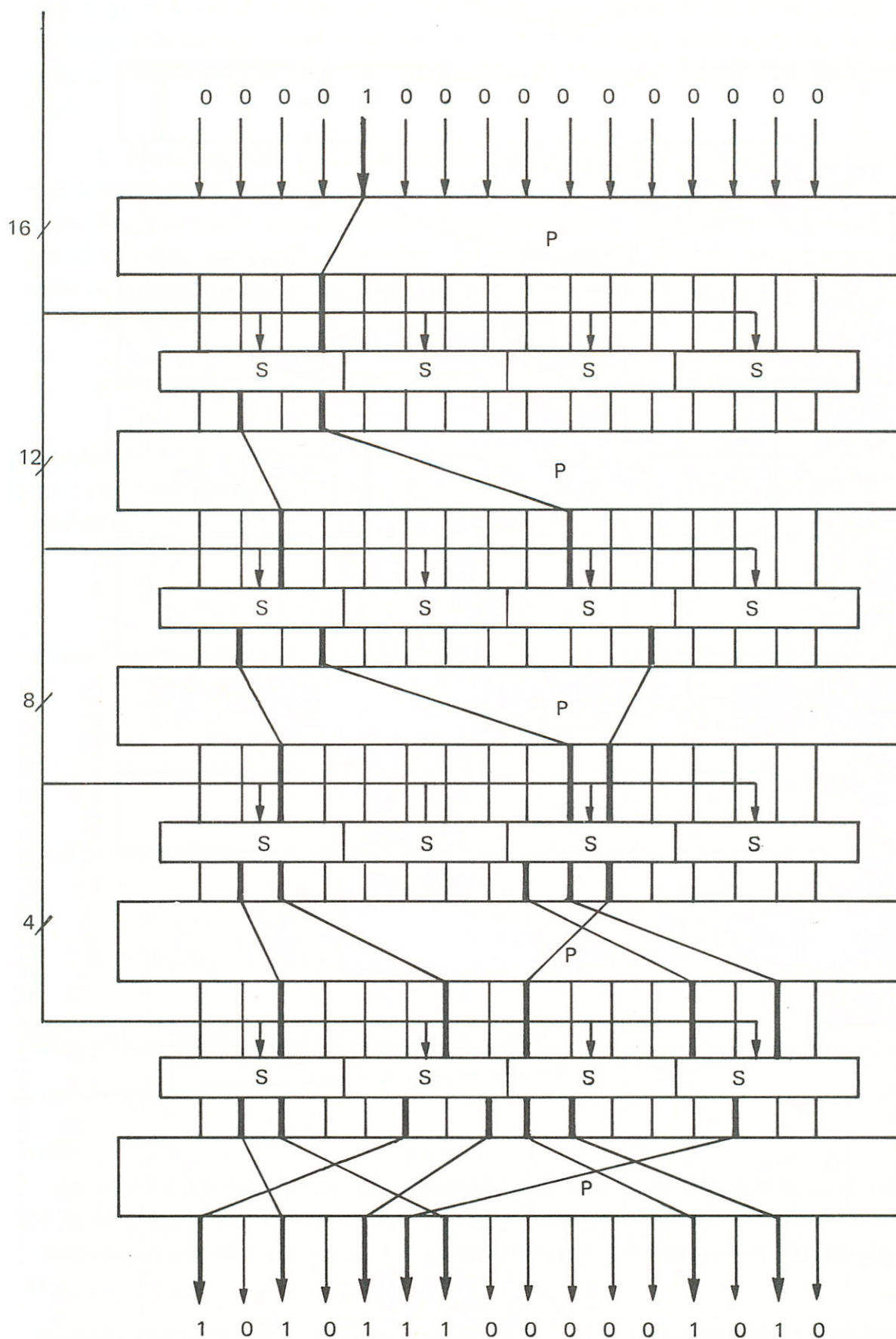
ún. **keverő transzformációként** hasznosította, amely váltakozó helyettesítési és transzpozíciós lépések egymás utáni alkalmazásából áll.

A keverő transzformáció lényegét és működési mechanizmusát a 3.1 ábra szemlélteti. Az áttekinthetőség végett 16 bites blokkhosszt és 16 bites kulcsot feltételezünk. A gyakorlatban mindkét hosszak az 50–150 bit tartományba kell esnie, hogy szisztematikus kísérletezéssel gazdaságosan vagy időben ne juthassunk el a transzformáció megfejtéséhez. A teljes transzformáció **rögzített permutációk** (transzpozíciók) és **kulcsvezérelt helyettesítések** több – esetünkben négy – rétegben való alkalmazásával épül fel. Az egyes permutációk különbözőek lehetnek, bár mindegyiket azonosan, P -vel jelöltük. Az egyes helyettesítési rétegekben a helyettesítés egymástól függetlenül 4 bites egységekben **S-dobozokban** történik. (Az egyes S-dobozok is különböző helyettesítéseket tartalmaznak.) A rejtést alapvetően a helyettesítések valósítják meg, a permutációk feladata, hogy egy helyettesítési lépés (réteg) eredményét „**szétszórják**” a következő helyettesítési réteg bemenetei között. (Egy-egy 4 bites helyettesítés eredménybitjeit a következő réteg különböző S-dobozaihoz vezetik.) Ez, mint az ábra is érzékelteti, a nyílt és rejtjelezett blokk bitjei közötti kölcsönhatást „**lavinyszerűen**” fokozza. Az egyes rétegekben egyetlen bemeneti bit értékének megváltozása egyre több és több bit értékét befolyásolja. Végeredményben minden nyílt blokkbeli bit megváltozása a rejtjelezett blokk bitjeinek közel felét érinti. (Ez teszi lehetetlenné a statisztikai alapú támadást.) Mindegyik S-doboz két helyettesítést valósít meg, amelyek között a kulcs odavezetett bitjének két lehetséges értéke választ. A lavinahatás miatt minden egyes kulcsbit is részt vesz a rejtjelezett blokk minden egyes bitjének meghatározásában, és egyetlen kulcsbit megváltozása is a rejtett blokk bitjeinek átlagosan a felét érinti.

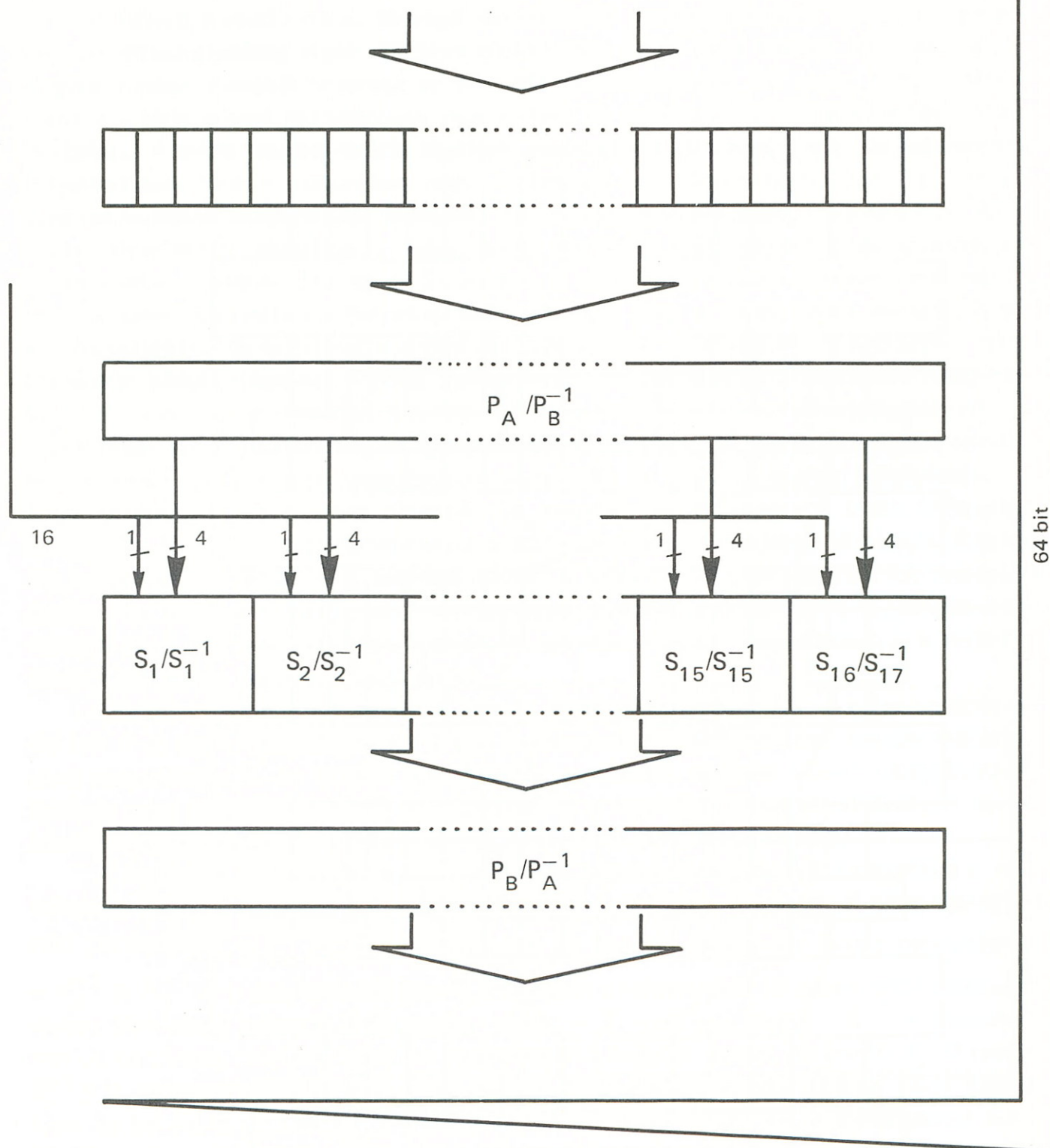
Könnyen felfedezhető, hogy ha azonos S-rétegeket választunk, a teljes transzformáció két P és egy S-rétegből, továbbá egy visszacsatolásból álló rendszer iteratív működésével megvalósítható. Az iteráció egyes lépései az eredeti, egymást követő rétegpároknak felelnek meg, és az iteráció minden lépésében is a megfelelő rétegben alkalmazott kulcsbitek használatát. A 3.2 ábra egy gyakorlati rendszer vázát mutatja.

A blokkhossz 64 bit, az egyes lépések 64 bites részeredményeit átmenetileg a vonalkódolt tároló tartalmazza. A megfordíthatóság érdekében mindegyik P réteg egy-egy direkt permutációt – P_A és P_B – valósít meg az elrejtés, és egy-egy inverz permutációt – P_A^{-1} és P_B^{-1} – a megfejtés céljára. Az S-rétegben a helyettesítés 16 egyenként 4 bites egységben történik. Mindegyik S-doboz egy direkt – S_i – és egy inverz – S_i^{-1} – transzformációt tartalmaz, amelyek között a hozzávezetett kulcsbit választ. (1. esetén S_i , 0 esetén S_i^{-1} .) A kulcs szintén 64 bites, és az iteráció 16 lépésének mindegyikében 16-ot használunk fel belőle. (Egyszeres felhasználás esetén minden kulcsbit 4-szer kerül sorra.) Az egy iteráción belül felhasznált kulcsbitek **iterációs kulcsnak** nevezzük, az iterációs kulcsok sorrendje a **kulcsütemezés**.

Rejtéskor a kulcsot eredeti formájában alkalmazzuk, és a direkt permutációkat – P_A és P_B – működtetjük. Fejtéskor az inverz permutációkat és a **kulcs invertált** formáját használjuk, az iterációs kulcsokat pedig **fordított sorrendben** ütemezzük. (A fejtés iterációs kulcsai tehát rejtéskor alkalmazottak inverzei, alkalmazási sorrendjük pedig eredeti sorrendjük fordítottja.) Az inverz permutációk, az inverz kulcsbitek miatti inverz helyettesítések alkalmazása és a fordított sorrendű iterációs kulcsütemezés következtében fejtéskor valóban a rejtés inverzét hajtjuk végre.



3.1 ábra.



3.2 ábra.

A teljes rejtjelezési transzformáció ellenállóképessége nyilvánvalóan az S -réteg helyettesítési transzformációjának tulajdonságaitól függ. Az S -dobozok kialakítását tehát körültekintően kell elvégezni. A választható helyettesítéseknek azonban számos megkötést kell kielégíteniük.

1. **Invertálhatóság:** Mint bemutattuk, a teljes transzformáció megfordíthatóságához kell.

2. **Bemenet-kimenet távolság:** A lavinahatás segítésére egy S -doboz bármelyik bemeneti bitkombinációjában egyetlen bit megváltozásának a kimeneten legalább két bit megváltozását kell eredményeznie.

3. **Méret:** Egy k -bités S -doboz $2^k \times 2^k$ bit (ROM) tárolókapacitással valósítható meg. Egy 4 bites S -doboz tehát 128, a 8 bites 4096, a 16 bites pedig 2097152 bit tárolókapacitás-igényű. Mivel egy rendszert 8–16 S -dobozból építünk fel, és egyetlen LSI integrált áramkörként akarunk megvalósítani, az S -dobozok jelenlegi felső mérethatára 8 bit.

4. **Nemlinearitás:** Az S transzformációknak a rejtjelfejtési kísérletekkel szembeni ellenállóképesség érdekében nemlineárisaknak kell lenniük, és **semmiképpen sem lehetnek** affin (lineáris plusz egy állandó) **transzformációk**. Affin S transzformációk eredménye ugyanis a teljes leképezés affinitása. (A permutációk lineáris transzformációk, az affin és lineáris transzformációk ötvözése ismét affin transzformáció.) A nyílt blokkot képviselő b és a rejtett blokkot képviselő c bináris vektorok között ilyenkor

$$c = A_k b + r_k$$

a kapcsolat. Bár a k kulcs meghatározása nehéz lehet, a behatoló megfelelő $n+1$ darab (b, c) pár birtokában könnyen hozzájuthat az A_k , A_k^{-1} és r_k értékekhez, amelyekkel a megfejtést

$$b = A_k^{-1} (c - r_k)$$

módon elvégezheti. Ehhez a $(b_0, c_0), (b_1, c_1) \dots (b_n, c_n)$ párokat normalizáljuk

$$b'_i = b_i - b_0$$

$$c'_i = c_i - c_0$$

majd ezeket oszlopvektorként használva kialakítjuk a

$$C = [c'_1 \ c'_2 \ \dots \ c'_n]$$

$$B = [b'_1 \ b'_2 \ \dots \ b'_n]$$

mátrixokat. Mivel a b'_i -k és c'_i -k kapcsolata

$$c'_i = A_k b'_i$$

ezért

$$C = A_k B$$

és

$$A_k = CB^{-1}$$

végül

$$r_k = c_0 - A_k b_0$$

128 bites blokkhosszúság esetén A_k és A_k^{-1} meghatározásához egyaránt közelítőleg $(128)^3$ művelet szükséges. Ennyi művelet végrehajtása $1\mu s$ utasításidő mellett csupán 2 mp rejtjelfejtési idő ráfordítást kíván. A tanulság kézenfekvő: az affin S transzformációk alkalmazását kerülni kell. Ez kizárja a 2 bites transzformációkat, mert ezek mindegyike affin.

E feltételek együtt jelentősen megnehezítik az S -dobozok transzformációinak kiválasztását. A szóban forgó transzformációk körét leginkább az invertálhatóság szűkíti. (n -bites transzformáció $(2n)^{2n}$ darab van, ebből azonban csak $(2n)!$ invertálható). Az invertálhatósági megkötés feloldása jelentheti tehát a legnagyobb könnyítést. Az IBM-nél dolgoztak ki olyan rejtjelezési eljárást, amelyben a helyettesítési transzformációnak nem kell invertálhatónak lennie. Ezt a következő fejezetben ismertetjük.

4. Az IBM rejtjelező rendszerei

Az IBM-nél a 60-as évek közepén bankterminálok védelmi rendszerének részeként kezdtek el a kor technológiai színvonalán megvalósítható rejtjelező berendezés kidolgozásával foglalkozni. Horst Feistelnek, aki ezeket a munkákat összefogta, sikerült olyan keverő transzformációt találnia, amely nem igényli a kulcsvezérelt helyettesítési transzformációk invertálhatóságát. Ezzel a szóba jöhető transzformációk száma jelentősen megnövekedett, és könnyebbé vált a helyettesítésre vonatkozó egyéb megkötések érvényesítése. Az elkészült berendezés a LUCIFER nevet kapta, és 128 bites blokkokat rejtjelezett 128 bites kulcs segítségével. A felhasznált algoritmus lényeges jellemzője, hogy a rejtés és fejtés ugyanazzal a berendezéssel történhet, amelynek vezérlése a transzformáció iránya szerint csak jelentéktelen mértékben, illetve formában változik. (Az akkor számításba vehető áramkör-technológia miatt fontos volt, hogy a berendezés minél kevesebb elemből álljon.)

A LUCIFER teljes kidolgozásához vezető kutatómunka körülbelül 6–7 évet vett igénybe. Amikorra elkészült, már feltűnt a látóhatáron a nagy integráltságú (LSI) áramkörök technológiája. Ezzel egyidejűleg a számítógépes távfeldolgozó rendszerek elterjedése, a számítógép-hálózatok megjelenése a rejtjelezés alkalmazása iránti igény rohamos növekedését helyezte kilátásba. Az IBM a LUCIFER nyomdokain haladva megindította azt a fejlesztési munkát, amelynek célja az LSI áramkörök lehetőségeihez igazodó rejtjelező berendezések kidolgozása volt. Az eredmény: 64 bites blokkokat 56 bites kulccsal rejtjelező algoritmus, amely egyetlen LSI áramkörrel készülékezhető. (A kulcs és blokkméret csökkentése ez utóbbi igénnyel függ össze.) Ezt az algoritmust az Egyesült Államok Szabványügyi Hivatala (NSB = National Bureau of Standards) 1977. január 15-i hatállyal (FIPS–PUB–46 szabvány) szabvánnyá nyilvánította. (A szabványosításhoz vezető felhasználási ajánlatot az IBM tette.)

A rejtjelezési szabvány, a DES (= Data Encryption Standard) hatásköre állami megrendelésre (vagy annak részeként) készült számítógépes rendszerekre terjed ki, amelyek – egyéb, katonai jellegű rejtjelezési szabvány hatáskörébe nem tartoznak;

- rejtjelezés alkalmazását igénylik;
- még nincs bennük rendszerbe épített rejtjelezési eljárás.

Szabványosnak azonban csak az algoritmus hardverbeli megvalósítása számít. A berendezések prototípusait be kell mutatni az NBS-nek, amely működésüket megfelelő vizsgálatokkal hitelesíti.

Bár az állami rendszerek körén kívül a szabvány alkalmazása nem kötelező, ténylegesen mégis szabványként működik, mert az integrált áramköröket gyártó cégek ezt az algoritmust megvalósító áramkört vagy áramkörkészletet gyártják és forgalmazzák. Ezen-

kívül egyéb szervezetek (felhasználói csoportoké, berendezésgyártóké stb.) fokozatosan saját ajánlásukként is elfogadják. A DES így ma világszabványnak tekinthető.

A fejezet további részében mindkét algoritmust ismertetjük. Az egyszerűbb tárgyalhatóság kedvéért azonban elsőként a DES-t mutatjuk be. Az egyébként azonos elvű és algoritmusú LUCIFER-ben ugyanis az áramkörüi kötöttségek az algoritmus lényegét kissé elfedik.

4.1 DES (Data Encryption Standard)

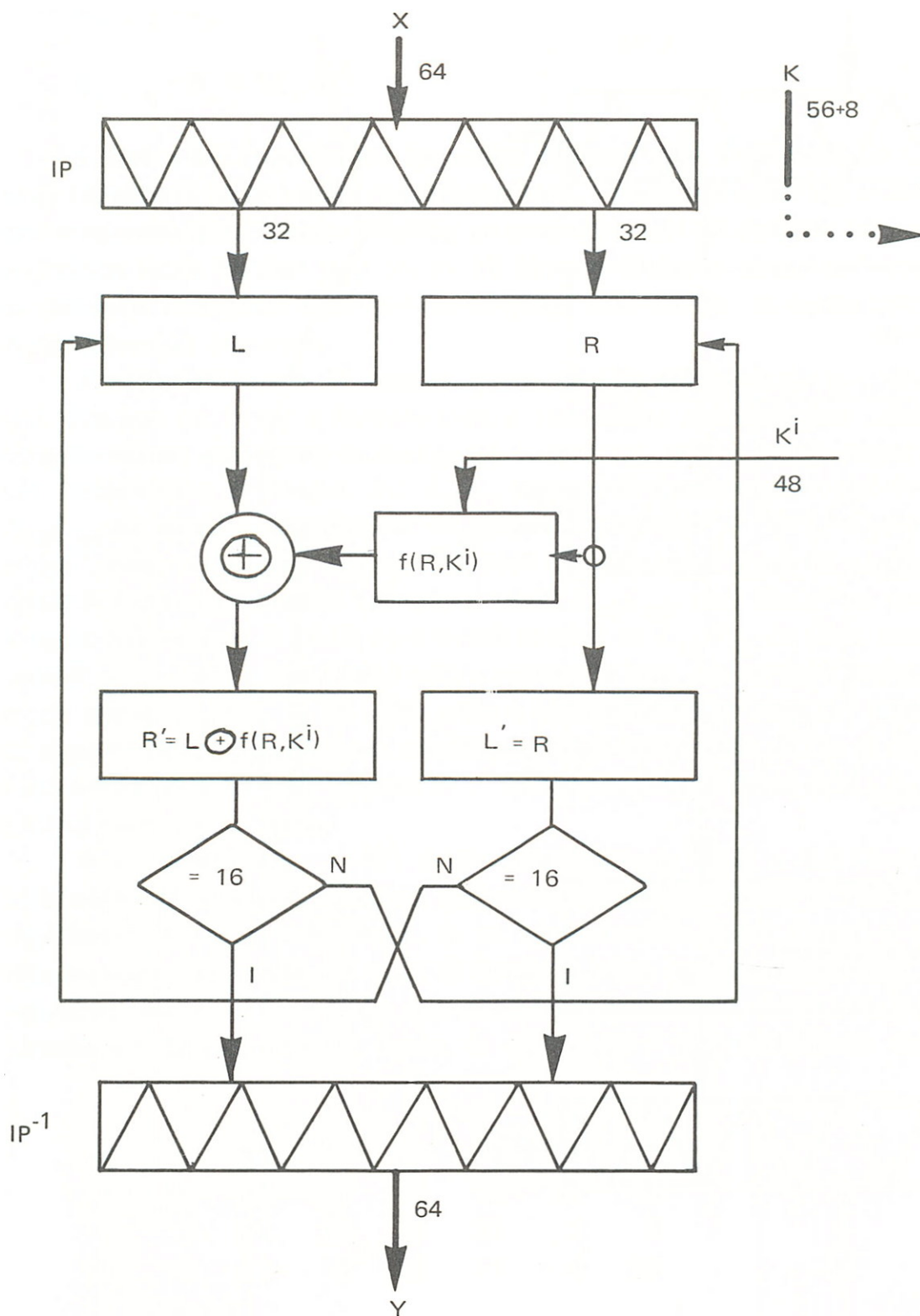
A DES algoritmus a már tárgyalt blokkrejtő eljárások általános szerkezetét követve ciklikus (4.1 ábra). A teljes transzformáció 16 azonos lépésből áll, amelyek mindegyikében a megfelelő, a rejtjelező kulcsból származtatott, **iterációs kulcsot** alkalmazunk. Az iterációs kulcsok előállítását **kulcsbitütemezésnek**, az iterációs kulcsok sorrendjét **kulcsütemezésnek** nevezzük. Az egyes iterációs kulcsokat az eredeti kulcs bitjeinek permutálásával és válogatásával származtatjuk. Mint látni fogjuk, a megfejtés csupán abban különbözik, hogy azonos kulcsbitütemezés mellett a kulcsokat **fordított sorrendben ütemezzük**. (Az i -edik iterációs lépésben K^i helyett a K^{17-i} -t alkalmazzuk.) A blokkméret 64 bit, és a K kulcs is 64 bitet tartalmaz, de ebből minden 8-adik bit paritásbit, az előző bit paritását páratlanra egészíti ki. A kulcs valójában tehát csak 56 bites.

Az adatblokkot a további feldolgozás előtt az IP **kezdeti permutációnak** vetjük alá.

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

tehát a permutáció után az eredetileg 58-adik bit válik elsővé, az 50-edik másodikká stb. Végül az eredetileg 7-edik bit lesz az utolsó. Ezt a permutációt követi az algoritmus 16 lépéses törzse, majd ennek eredményére a kezdeti permutáció inverzét alkalmazzuk.

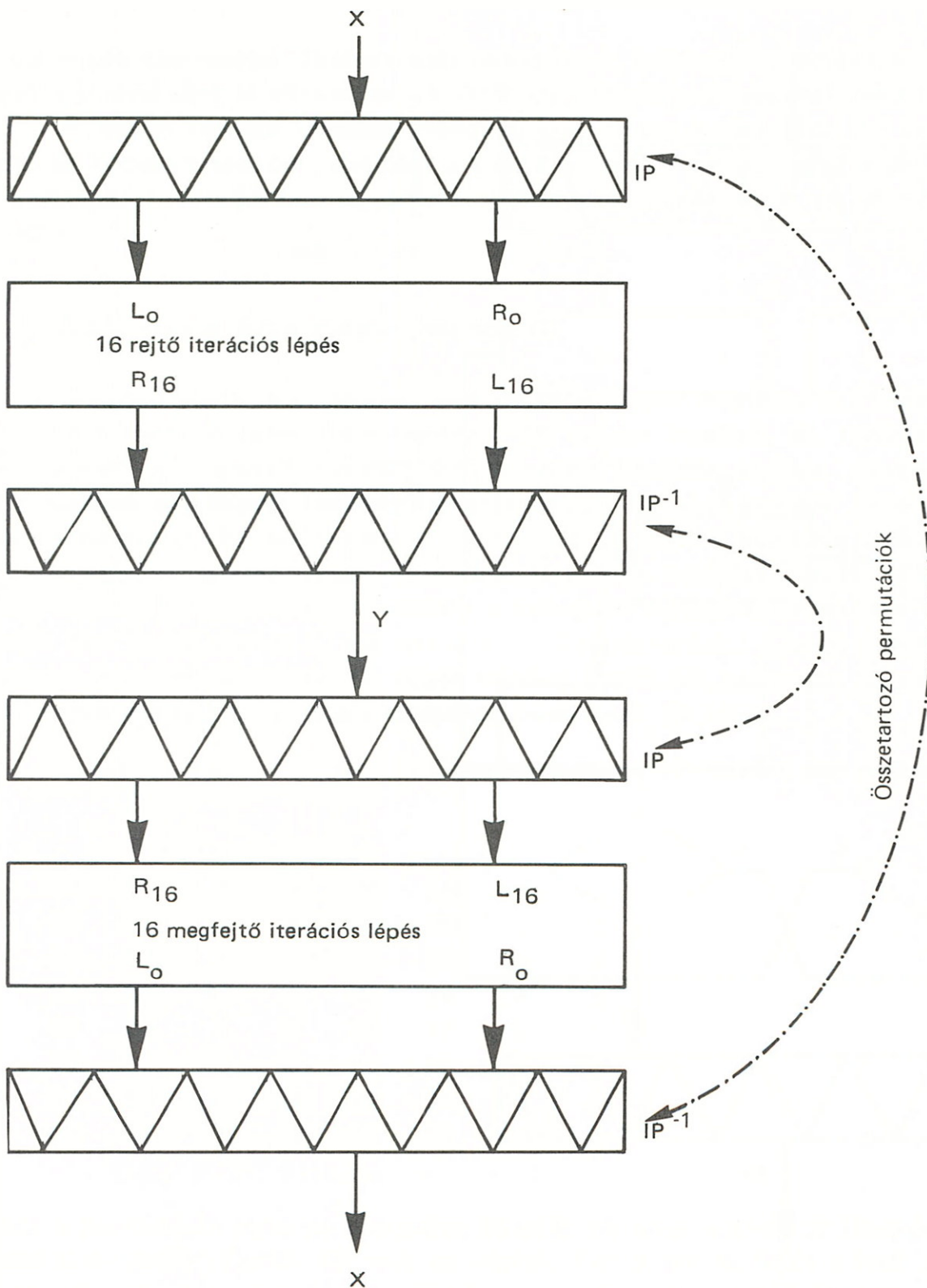
IP ⁻¹							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



4.1 ábra.

Vagyis a transzformáció eredményének 40-edik bitje lesz a kimeneti blokk első bitje, a 8-adik kerül a második pozícióba stb. Végül a 25-ödik bit kerül a kimeneti blokk végére. Az IP és IP^{-1} transzformációk szerepét később még egyszer szemügyre vesszük.

Az iteratív transzformáció minden egyes lépése a pillanatnyi 64 bites blokkot két 32 bites félblokként kezeli. Egy iteratív lépés során a jobb oldali félblokk – R_{i-1} – nem változik. A bal oldali félblokkhoz – L_{i-1} -hez – viszont mod-2 hozzáadjuk a jobb oldali félblokk és az iterációs kulcs 32 bitet eredményező függvényét, $f(R_{i-1}, K^i)$ -t. A két így kapott félblokk a következő iterációban szerepet cserél, a bal oldaliból jobb oldali, a jobb oldaliból bal oldali félblokk lesz. Ez alól csak az utolsó iteráció kimeneti ered-



4.2 ábra.

ménye kivétel, a végső permutációhoz a félblokkok csere nélkül kerülnek. Ha tehát a (permutált) bemenet $L_0 R_0$, akkor az iterációk végén a (permutálatlan) kimenet $R_{16} L_{16}$.

Az i -edik transzformációs lépés eredménye

$$\left. \begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K^i) \end{aligned} \right\} \quad (4.1)$$

Az egyes iterációs kulcsokat pedig a **kulcsütemező** állítja elő. Adjuk a második egyenlet mindkét oldalához $(R_i \oplus L_{i-1})$ -et, és helyettesítsük benne R_{i-1} -et L_i -vel. Ekkor

$$\left. \begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f(L_i, K^i) \end{aligned} \right\} \quad (4.2)$$

Ennek a két egyenletnek igen nagy a jelentősége. Értelmezésük ugyanis azt mutatja, hogy bármelyik lépés hatása megszüntethető, ha eredményére ugyanazzal az iterációs kulccsal megismételjük, miután előzőleg az eredmény két félblokkját felcseréltük. A 16 transzformációs lépés hatását egy olyan 16 lépéses újabb ciklussal szüntethetjük meg, amelyben az iterációs kulcsokat fordított sorrendben ütemezzük. A rejtés-fejtés kapcsolatot a 4.2 ábrán követjük nyomon.

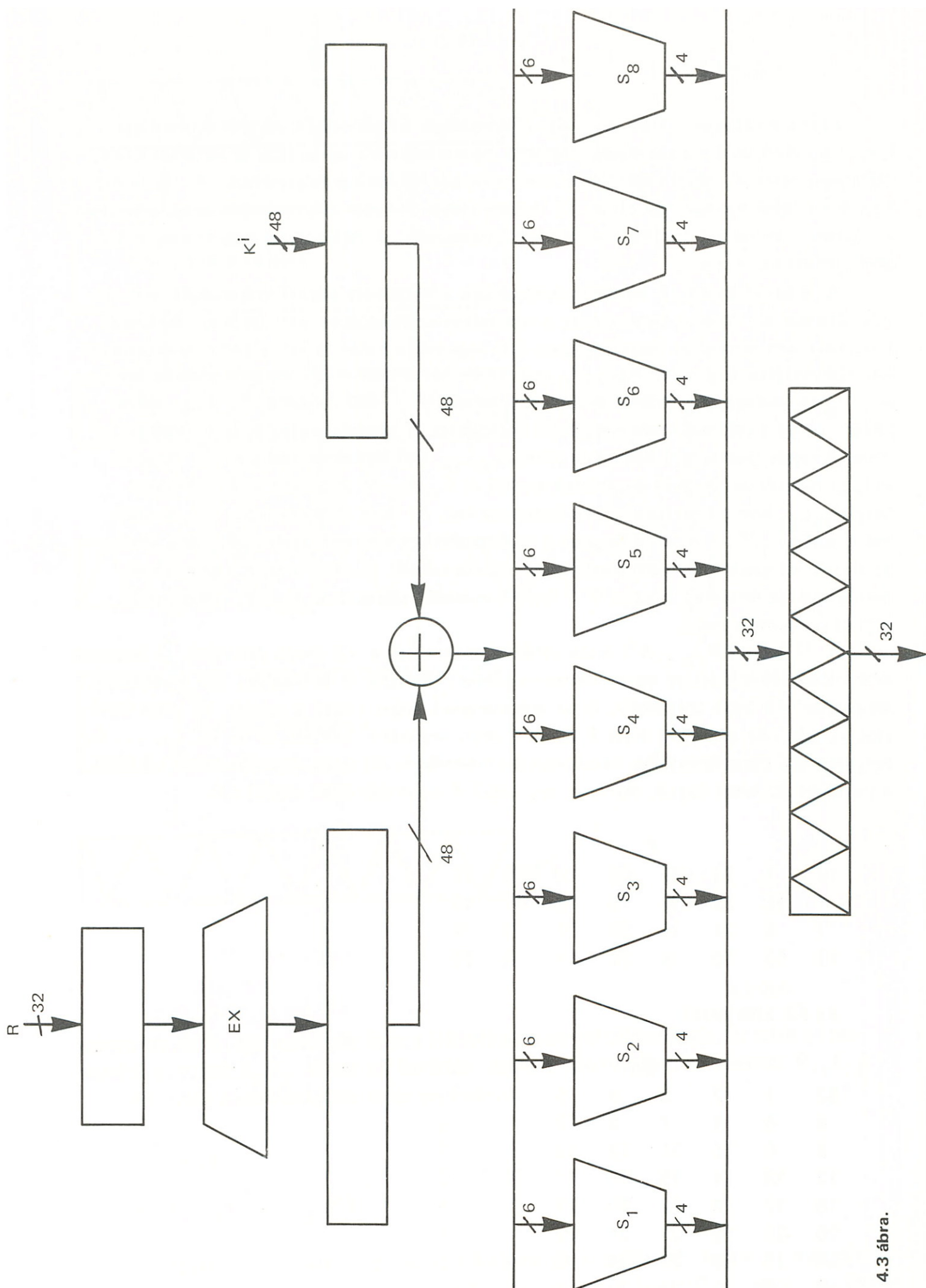
A rejtendő blokk IP kezdeti permutációja a rejtjelezés ellenállóképességét nem növeli. Szerepe az, hogy a bemenetnek a redundáns számítógépes adat (szöveg) ábrázolás következtében eredetileg összefüggésben álló (egymásra ható) biteit a permutált bemenet két félblokkjára szétossza. Az L_0R_0 permutált bemenetet a 16 transzformációs lépés $R_{16}L_{16}$ -ba alakítja. Az ábráról ezután leolvasható IP^{-1} első feladata: $R_{16}L_{16}$ „előtorzítása”, hogy a rejtéssel azonos megfejtő berendezés IP kezdeti permutációját hatástalanítsa. A fejtés (permutált) bemenete tehát $R_{16}L_{16}$, első lépéséhez pedig a K^{16} iterációs kulcsot használjuk. A (4.2) egyenletek szerint ez $L_{15}R_{15}$ -öt adja, amely félblokkcsere után $R_{15}L_{15}$ -ként a következő ciklus bemenete lesz. Az iterációs kulcs most K^{15} , az eredmény pedig $L_{14}R_{14}$. A 16 lépés után a (permutálatlan) kimenet tehát L_0R_0 , amelyben az algoritmus szerint már nem alkalmazunk félblokkcsere-t. (Most és a fejtés kezdetén látható ennek az értelme.) Ezután IP^{-1} betölti második szerepét: L_0R_0 -t az eredeti nyílt blokká permutálja vissza.

A 32 bites $f(R_{i-1}, K^i)$ érték előállításának vázát a 4.3 ábrán láthatjuk. A bemenet jobb oldali félblokkját az egyes bemeneti biteket többször is felhasználó **EX kiterjesztés** segítségével 48 bitté szélesítjük, és az eredményhez adjuk hozzá a szintén 48 bites iterációs kulcsot. Az eredő 48 bitet 8 darab 6 bites blokként kezeljük, és az $S_1, S_2, \dots, S_8$ helyettesítési transzformációk bemeneteként használjuk. Az egyes transzformációk 4 bites kimeneteit 32 bitté fogjuk össze, és egy végső P permutációnak vetjük alá.

P							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Az EX kiterjesztés

EX					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1



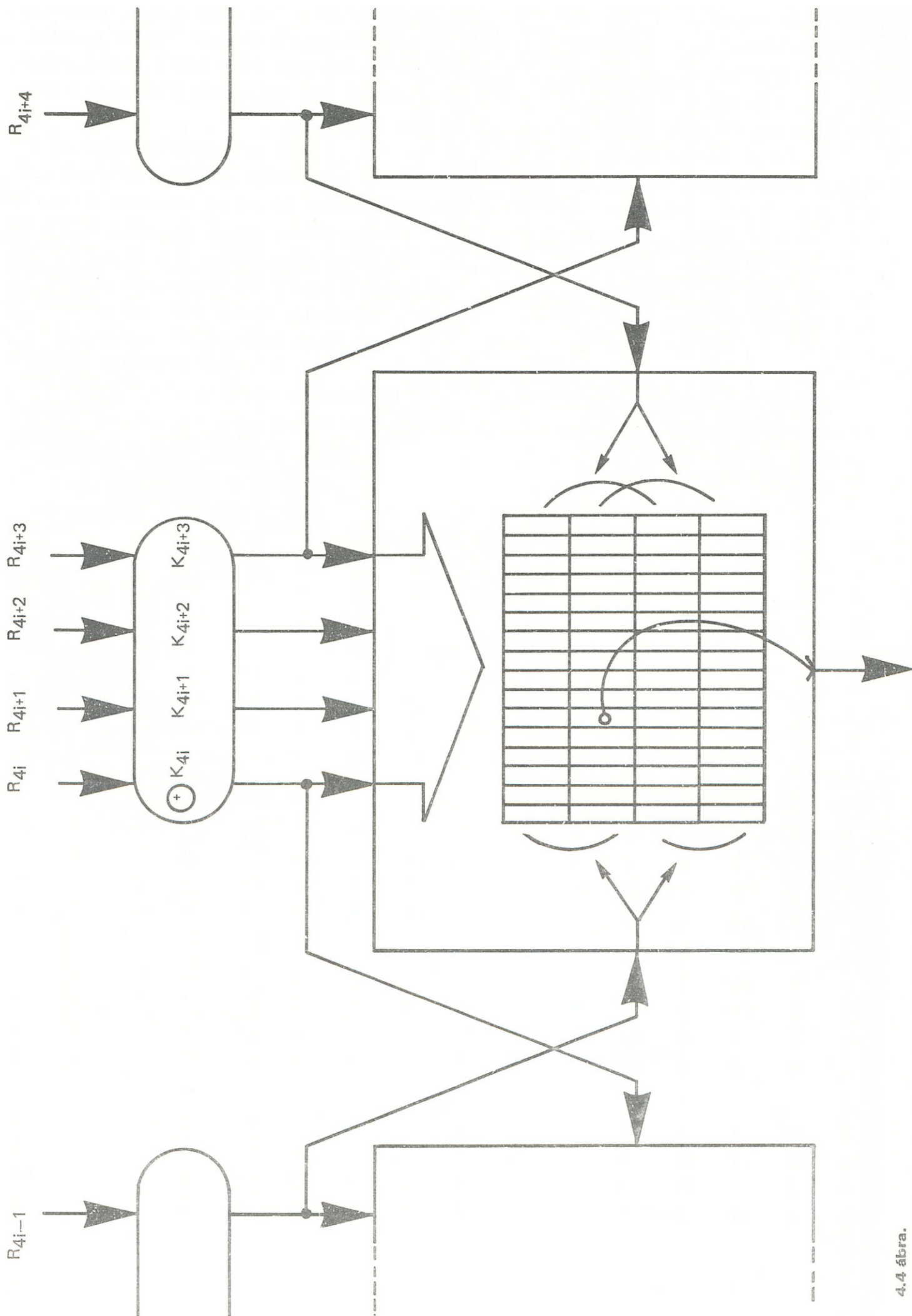
4.3 ábra.

(A táblázat R -beli bitpozíció-sorszámokat tartalmaz, $EX(R)$ bitjei sorfolytonos „összeolvással” adódnak.) A kiterjesztéshez tehát R -t 4 bites blokkokra osztjuk. Ezeket egyenként 6 bitté terjesztjük ki úgy, hogy minden blokk elé a ciklikusan előző blokk utolsó, mögé pedig a ciklikusan következő blokk első bitjét csatoljuk. Az így kapott 8 darab 6 bites blokk alkotja $EX(R)$ -t, illetve az S_i helyettesítések bemenetét.

A 4.4 ábrán ennek megfelelően részleteztük $f(R_{i-1}, K^i)$ előállításának egyetlen S -dobozt érintő részét. Minden S -doboz 4, egyenként 4 bites helyettesítő transzformációt tartalmaz. A doboz ezeket 4, egyenként 16 elemű táblában tárolja. A bemeneti 6 bites blokk első és utolsó bitje (ebben a sorrendben) 2 bites számot alkot, amelynek numerikus értéke választja ki az alkalmazandó táblát. A kiválasztott táblának a középső 4 bittel megcímezett eleme foglalja magában a helyettesítési értéket. A kiterjesztés, ilyen megválasztásával, tehát segíti a korábban már említett lavinahatás kialakulását. Ehhez természetesen a P permutáció, valamint a bal oldali és jobb oldali félblokkok cserélgetése is hozzájárul.

Végül az S -dobozok tartalma a következő: (Minden sor egy 4 bites helyettesítési transzformáció, ez egyes 4 bites kombinációkat numerikus értékükkel adtuk meg, és a bal szélső oszlop címe 0.)

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3



4.4 ábra.

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	11	7	6	0	8	13

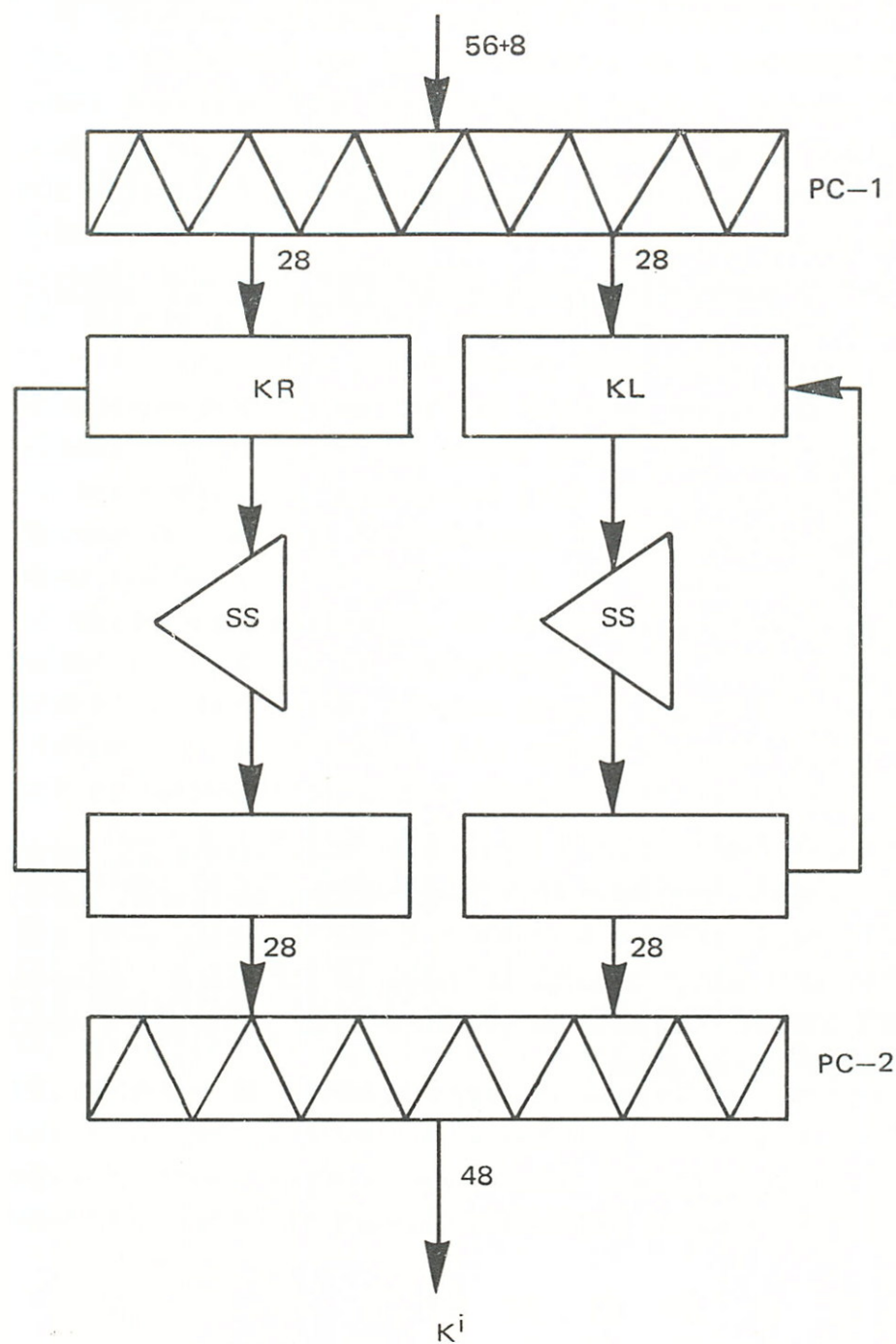
S_6

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_7

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S_8



4.5 ábra.

A DES teljes definíciójához meg kell még adnunk a kulcsbit- és kulcsütemezés részleteit is. A 4.5 ábra szerint az eredeti kulcs bitjei elsőként egy *PC-1* permutált kiválasztáson mennek keresztül.

<i>PC-1</i>						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Ez, mint látható, a kulcsbitek permutációjával egyidejűleg kiszűri a paritásbiteket. A kapott 56 bitet két 28 bites félként (*KR*, *KL*) kezeljük, és külön-külön minden iterációs lépés előtt ciklikusan balra körbeléptetjük. Az egyes ciklusokban alkalmazandó lépések számát *SS* határozza meg.

	<i>SS</i>															
iteráció:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
léptetés:	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Az eredő $2 \times 28 = 56$ bitből *PC-2* permutált kiválasztással állítjuk elő az iterációs lépés 48 bites kulcsát.

<i>PC-2</i>					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Vegyük észre, hogy a 16 iterációs lépés alatt a léptetések teljes száma 28, tehát a rejtjelezés végén visszakapjuk az eredeti, paritásbitjeitől megfosztott, permutált kulcsot: $KL_{16} = KL_0$, $KR_{16} = KR_0$. A kulcsütemezés tehát a következő blokk fogadása előtt már alaphelyzetbe kerül. Fejtéskor mindössze annyi a változás, hogy az *SS* táblát „visszafelé olvassuk”, és balra forgatás helyett jobbra forgatást alkalmazunk. (Ez ugyanazt a kulcsbitütemezést eredményezi, de fordított sorrendben.)

4.2 LUCIFER

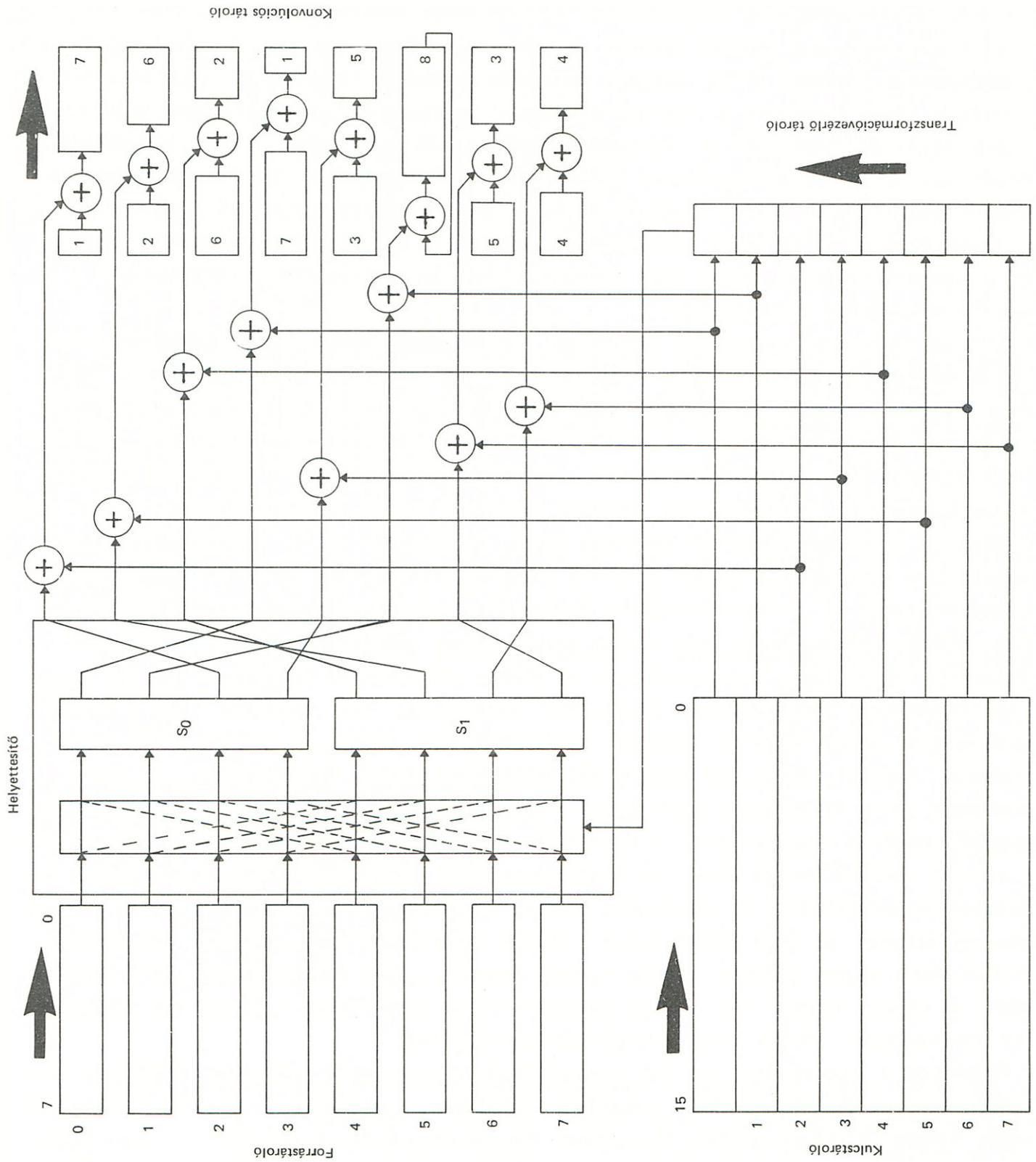
A DES „ősatyjának” — a LUCIFER-nek — mint berendezésnek a felépítését és működését a 4.6 ábra segítségével vizsgálhatjuk. A 128 bites adatblokk és a szintén 128 bites kulcs bitjeit egyaránt léptető tárolók tartalmazzák. Egy adat vagy kulcsoktett bitjei nyolc **párhuzamos** léptető tároló **azonos sorszámú pozícióit** foglalják el. (Az ábra szerint az oktettek „függőlegesen”, a léptető tárolókon „keresztben” helyezkednek el.) Az adatblokk egyik 64 bites fele a **forrástárolóban**, másik fele pedig az un. **konvolúciós tárolóban** van. Ezeknek a tárolóknak mindegyike körbeléptethető, ezenkívül tartalmuk léptetéssel felcserélhető, illetve kívülről tölthető vagy együttes tartalmuk kiléptethető. (A mozgást függőleges oktettek mozgásaként képzeljük el.) Az áttekinthetőség végett ezeket a kapcsolatokat az ábrán nem tüntettük fel, de a működés ismertetésekor a megfelelő műveletekre hivatkozunk.

A teljes működés a DES-hez hasonlóan 16 **iterációs lépést** foglal magában, minden iterációs lépés további 8 fázisból áll, és a forrás, valamint a konvolúciós tárolók tartalma minden iterációs lépés végén helyet cserél. Egy iterációs lépésben a forrástároló tartalma változatlan marad. A konvolúciós tároló tartalma viszont eredeti tartalmának, a kulcsbitek egy részhalmazának és a forrástároló tartalmára alkalmazott kulcsvezérelt transzformáció eredményének mod–2 összege lesz. (A forrástároló tartalma a DES jobb oldali, a konvolúciós tároló tartalma pedig a DES bal oldali félblokkjának szerepét játssza.) A konvolúciós tároló tartalma azonban az iterációs lépés 8 fázisa során **fokozatosan** alakul ki. Minden iterációs lépés elején a **transzformációvezérlő** (szintén léptető) **tárolót** a **kulcstároló** jobb szélső pozíciójában található kulcsoktettel töltjük fel. Ezután a nyolc fázis mindegyikében az alábbi műveleteket hajtjuk végre.

A forrástároló pillanatnyilag jobb szélső oktettségének két négybites felét a transzformációvezérlő tároló legfelső bitjének megfelelően (ha értéke 1) megcseréljük, vagy változatlanul hagyjuk. Az eredő oktettség két felét külön-külön helyettesítjük (S_0 és S_1). Az eredmény és a kulcstároló jobb szélső oktettségének azonos sorszámú bitjeit mod–2 összeadjuk, majd az összegbitek a konvolúciós tároló más-más oktettségének egy-egy bitjéhez szintén mod–2 hozzáadjuk. (A kiválasztott bitet a mod–2 összegezőtől balra levő tárolóelem tartalmazza, az eredmény a mod–2 összegező kimenetén alakul ki. A tárolókba írt számok az összeadótól balra, illetve jobbra található tárolóelemek számát jelölik.) A fázis végén minden tárolót eggyel léptetünk, a transzformációvezérlő tárolót „felfelé”, a többi jobbra. (Az előbbi összegek ekkor kerülnek a konvolúciós regiszternek az összeadóktól jobbra elhelyezkedő tárolóelemeibe.)

Rejtéskor a kulcstárolót az iterációs lépések utolsó fázisa után **nem léptetjük**, fejtés esetén viszont ilyenkor **kétszer léptetjük**. Fejtéskor a kulcstároló alapállapotát betöltés utáni kilenc (ciklikus) jobbra léptetéssel állítjuk be. Ezek a szabályok a 4.1 táblázat szerinti kulcsoktett-ütemezéshez vezetnek. Láthatóan rejtéskor és fejtéskor az iterációs lépésekben ugyanazokat a kulcsoktetteket használjuk, csak fordított sorrendben. (A transzformációt egy iterációs lépésben az elsőnek ütemezett kulcsoktett bitjei vezérlik, rejtéskor és fejtéskor egyaránt ugyanazon bitsorrendben.) A DES-nél elmondottak szerint mindez pontosan a berendezés megfejtésre való alkalmazásának szükséges és elégséges feltétele. Végül a helyettesítő transzformációk (a bemenet numerikus értéke a táblázatosan megadott transzformációkból indexeléssel választja ki a helyettesítési értéket):

S_0 :	12	15	7	10	14	13	11	0	2	6	3	1	9	4	5	8
S_1 :	7	2	14	9	3	11	0	4	12	13	1	10	6	15	8	5



4.6 ábra.

4.1 táblázat.

Ciklus	Rejtés								Fejtés							
0	0	1	2	3	4	5	6	7	9	10	11	12	13	14	15	0
1	7	8	9	10	11	12	13	14	2	3	4	5	6	7	8	9
2	14	15	0	1	2	3	4	5	11	12	13	14	15	0	1	2
3	5	6	7	8	9	10	11	12	4	5	6	7	8	9	10	11
4	12	13	14	15	0	1	2	3	13	14	15	0	1	2	3	4
5	3	4	5	6	7	8	9	10	6	7	8	9	10	11	12	13
6	10	11	12	13	14	15	0	1	15	0	1	2	3	4	5	6
7	1	2	3	4	5	6	7	8	8	9	10	11	12	13	14	15
8	8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	8
9	15	0	1	2	3	4	5	6	10	11	12	13	14	15	0	1
10	6	7	8	9	10	11	12	13	3	4	5	6	7	8	9	10
11	13	14	15	0	1	2	3	4	12	13	14	15	0	1	2	3
12	4	5	6	7	8	9	10	11	5	6	7	8	9	10	11	12
13	11	12	13	14	15	0	1	2	14	15	0	1	2	3	4	5
14	2	3	4	5	6	7	8	9	7	8	9	10	11	12	13	14
15	9	10	11	12	13	14	15	0	0	1	2	3	4	5	6	7

Kulcsoktett-ütemezés

4.3 A DES értékelése

A DES körül szinte azonnal szenvedélyes vita kerekedett, amint az NBS szabvány tervezetként napvilágot látott. A vitát és a DES-szel szembeni bizalmatlanságot az keltette, hogy az IBM a korábban használt 128 bites kulcshosszat 56-ra csökkentette, valamint a DES kidolgozásának és vizsgálatának részleteit tartalmazó anyagokat nem hozta nyilvánosságra. (Mindössze annyit közölt, hogy a DES 17 emberévnyi összpontosított rejtjelfejtési erőfeszítésnek sikeresen ellenállt.) A DES-tervezet ellenzői az 56 bites kulcshosszat túlságosan rövidnek találták ahhoz, hogy az állandó technológiai fejlődés eredményeként megvalósítható számítóteljesítmény-növekedés ellenére huzamos időtartamra (kb. 10 évre) megfelelő védeltséget nyújtson. A vizsgálati anyagok visszatartása miatt pedig valamilyen beépített csapdára gyanakodtak. Mivel a DES szabványtervezetének kidolgozásába hivatalból bekapcsolódott az amerikai hadügyminisztérium biztonsági ügynöksége, az NSA (National Security Agency) is, a DES módosításának követelői politikai „összeesküvést” véltek felfedezni. Martin E. Hellman — aki egyébként a rejtjelezési módszerek nyilvános kutatási területének kiemelkedő alakja — egyenesen így fogalmazott: „Miért támogat az NBS egy kellő biztonságot nem nyújtó szabványt, és miért hadakozik a javításai ellen? A válasz politikai, és azzal kapcsolatos, hogy az NSA hatáskörébe esik az Egyesült Államokban minden rejtjelkidolgozó és rejtjelfejtési tevékenység. Tömören fogalmazva: a biztonsági ügynökség nem akarja, hogy nyilvánosan hozzáférhető legyen egy olyan rejtjelezési rendszer, amelyet nem képes megfejteni. Bár a Szenátus Hírszerzési Különbizottsága a DES-t megfelelőnek minősítette, egyben úgy találta, hogy a DES kidolgozása során az NSA meggyőzte az IBM-et arról, hogy egy csökken-

tett kulcshossz is megfelelő.” Ezzel szemben az IBM magyarázatát Walter L. Tuchman tolmácsolja [16]: „Miután a kidolgozott rejtjelezési eljárást az NBS-nek szabványosításra átadtuk, az NSA felhívta a figyelmünket arra, hogy akaratlanul újra felfedeztük a rejtjelezési kutatás egyik fontos eszközét — abban az értelemben, hogy újra feltaláltuk néhány analízis és tervezési módszerüket.” A vitát hivatalosan a Szenátus Hírszerzési Különbizottságának vizsgálata zárta le, amelynek nyilvánosságra hozott következtetései a következők voltak:

- A DES a szándékolt alkalmazásainak minden szempontból megfelel.
- A DES-t az IBM találta fel és dolgozta ki.
- Az NSA nem változtatott a kidolgozott megoldáson.
- Az ügynökség (NSA) tanúsította, hogy a DES mentes minden ismert statisztikai vagy matematikai gyengeségtől.

A szakmai vita azonban sokkal lassabban ül el. Martin Hellman és munkatársai a DES potenciális gyengeségei után kutatva részletesen szemügyre vették a szerkezetét, és a viszonylag rövid kulcsból eredő sebezhetőségét. Elsőként a DES szerkezetét érintő véleményüket foglaljuk össze.

A DES legnagyobb szerkezeti gyengeségét egy szimmetriatulajdonsága jelenti. Az $f(R, K)$ függvény alakjából következik, hogy

$$f(R, K) = f(EX(R) \oplus K)$$

Mivel $EX(R)$ -t R bitjeiből állítjuk elő, R invertálásával együtt $EX(R)$ is invertálódik. R és K együttes invertálása esetén ezért $EX(R) \oplus K$ változatlan marad.

$$f(R, K) = f(\bar{R}, \bar{K})$$

Ezt a (4.1) egyenletekkel összevetve (a K' iterációs kulcsok K -val együtt invertálódnak) következik, hogy ha

$$C = E_K(B)$$

akkor a kulcs és a nyílt blokk együttes invertálásának eredménye a rejtett blokk invertálása is

$$C = E_{\bar{K}}(\bar{B})$$

Ez a szimmetriatulajdonság a **választott nyílt** szövegű rejtjelfejtés időigényét **felére csökkenti**. Tegyük fel ugyanis, hogy a behatoló (B, C_1) és $(\bar{B}, C_2)$ nyílt-rejtett blokk-párokkal rendelkezik. Az alkalmazott kulcs keresésekor azokkal a kulcsokkal rejti B -t, amelyek legfelső bitje 0. Ha $E_K(B) = C \neq C_1$, akkor C -t összehasonlítja C_2 -vel is. Ha $C \neq C_2$, a fenti szimmetriatulajdonság miatt K sem lehet a keresett kulcs. Mivel egy összehasonlítás lényegesen gyorsabban hajtható végre, mint egy rejtés, a behatoló lényegében egyetlen rejtés időszükségletével egyszerre két kulcsot zárhat ki, a rejtjelfejtés időigénye felére csökken.

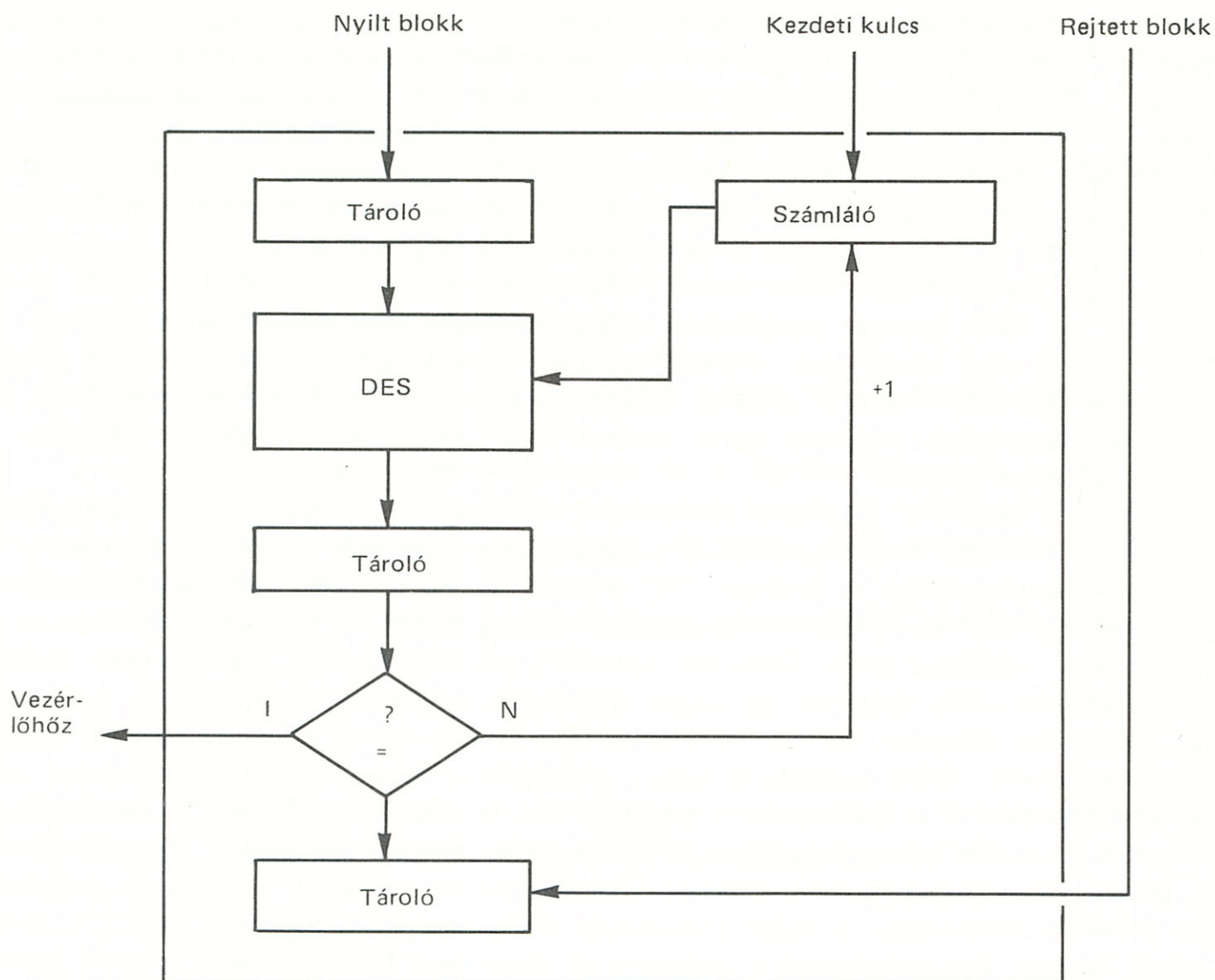
A másik kifogásolható megoldást a kulcs paritásbitek azonnali kiszűrése jelenti. Első pillantásra ugyan logikusnak tűnik, hogy a kulcsbitütemezés során a többi (szabadon választható) kulcsbittől lineárisan függő paritásbiteket ne használjuk fel, a részle-

tesebb vizsgálat azonban azt mutatja, hogy ennek nincs rejtjelezési jelentősége. Mivel az egyes iterációs kulcsok 48 bitesek, PC-2 megtervezhető úgy, hogy ne szerepeljen bennük egy-egy kulcsoktett összes bitje. Az iterációs kulcsok tehát a paritásbitek felhasználása ellenére is független kulcsbitekből állhatnak. A teljes kulcs felhasználása ugyanakkor választási lehetőséget adna a felhasználónak: ha a kulcs helyességének ellenőrizhetőségéről lemond, a rendelkezésére álló kulcstartományt $2^8 = 256$ -szorosára kiterjesztheti. Ez a szorzótényező, mint látni fogjuk, a rejtjelfejtés erőforrásigényében el nem hanyagolható különbséget eredményez.

Ezeket kívül lényeges rejtjelfejtési ráfordításcsökkentésre módot adó szerkezeti gyengeséget a DES bírálói nem találtak. Az S-dobozok leképezéseiben felfedezett néhány, lineárishoz közelálló (de nem lineáris) leképezésrészlettel kapcsolatban pedig az NSA kijelentette, hogy nincs tudomása olyan módszerekről, amelyekkel ezeket a rejtjelfejtésben hasznosítani lehetne.

Mivel a kétkedők szerkezeti gyengeségek formájában nem találtak hatásos segítséget a DES kikezdéséhez, figyelmüket a kulcs rövidségének kihasználási lehetőségeire összpontosították. Diffie és Hellman [12] érvelése az, hogy a DES születése pillanatában még a „nyers erőszakkal” — kimerítéses próbálgatással — szemben is csak éppen elfogadható védelmet nyújt. Szerintük ugyanis a jelenlegi technológiai fejlettség mellett is építhető 20 millió dollárért egy olyan rejtjelfejtő berendezés, amely ismert nyílt szöveg birtokában átlagosan fél nap alatt megtalálja az ismert nyílt-rejtett blokkpárt összekapcsoló kulcsot. Teljes terhelést és 5 éves amortizációt feltételezve megfejtésenként ez 2500 dollárt jelent. A berendezés nagyszámú, ún. kulcskereső párhuzamos működtetésével végzi feladatát. Egy-egy kulcskereső (4.7 ábra) lényegében egy DES rejtést valósít meg, és tárolni képes az adott nyílt blokkot, valamint rejtett megfelelőjét. A rejtés kulcsát egy számláló tartalmazza, a rejtés eredményét pedig állandóan összehasonlítjuk a tárolt rejtett blokkal. Egy kulcskereső a végignézendő összefüggő kulcstartomány kezdeti kulcsát indításkor kapja meg a kulcstartomány hosszával (a kipróbálandó kulcsok számával) együtt. Ezt követően ciklikusan működik, minden ciklus rejtést, összehasonlítást, majd kulcsléptetést tartalmaz. Ha összehasonlításakor egyezést talál, azt az egyezést kiváltó kulccsal együtt a rejtjelfejtő berendezés központi vezérlésének jelenti. (Pontosabban jelzi, amikor működéseről a központi vezérlő lekérdezéssel jelentést kér.) Egy kulcskereső áramkör kb. 6400 kapuból áll, így bonyolultsága megfelel egy Z80 mikroprocesszorénak. Megépíthető tehát egyetlen LSI áramkör formájában. Már kipróbált és bevált technológiák alkalmazásával 0,8–1 watt teljesítményfelvétel mellett (amely hűtés szempontjából szintén általánosan elfogadott érték) $1\mu s$ keresési ciklusidő érhető el. Folyamatosan működtetve tehát egy kulcskereső egyetlen nap alatt $8,64 \times 10^{10}$ kulcsot képes kipróbálni. Mivel a DES kulcstartományának mérete $2^{56} = 7 \times 10^{16}$, egy egymillió kulcskeresőből felépített rejtjelfejtő berendezés egyetlen nap alatt a teljes kulcstartományt kimerítheti, ha átfedések nélkül felosztja az egyes kulcskeresők között. Mivel a találathoz átlagosan a teljes kimerítés lépésszámának a fele elegendő, egy-egy rejtjelfejtési feladat megoldásának időigénye fél nap.

A kérdés most már csak az, hogy a rejtjelfejtő berendezés egyáltalán megépíthető-e, és ha igen, akkor hatékonyan üzemeltethető-e. Diffie és Hellman véleménye erről a következő: Mivel a kulcskeresők önállóan dolgoznak, B/K igényük alacsony, ezért soros B/K alkalmazható. Következésképpen a kulcskereső áramkör kivezetéseinek (lábainak) száma csekély, szerintük 7. Ez lehetővé teszi, hogy egy 28x33 cm-es nyomtatott lapra 128 kulcskereső és a rendszerbe illesztésükhöz szükséges csatoló elférjen. A rejtje-



4.7 ábra.

lező berendezés tehát 64 fiókból, fiókonként 128 ilyen kártyából felépíthető. Teljesítményfelvétele ugyan 2 MW, de ez nem jelent megoldhatatlan hűtési feladatot. A **hibátlan működés szabad úthossza** az LSI áramkörökre tipikusan megadott 0,05%/1000 óra meghibásodási arány mellett 2 óra, de a még lényegesen rosszabb 1%/1000 óra érték mellett is legalább 6 perc. Ha egy meghibásodott kártya cseréje 10 percet vesz igénybe, a rendszer folyamatos üzemeltetése még a legrosszabb esetben is biztosítható 2-3 tartalék fiók segítségével. Ehhez az egyes fiókok:

- saját meghibásodásukat észlelik,
- a meghibásodott kártyát kijelzik és
- tartalékra kapcsolásukkor a keresés folytatásához szükséges adatokat a központi vezérlőhöz továbbítják.

A működés ellenőrzése 30 percenként, a folyamatos működés megszakításával történik. Az ellenőrzés megkezdése előtt minden kártya csatolóegysége megjegyzi a lefutott ciklusok számát, majd minden fiók ugyanazt a 10 keresési feladatot kapja egymás után. Ezek mindegyike olyan, hogy 10 lépésben célhoz vezet. Ezzel a módszerrel az egyes kulcskeresők működésének helyességét a normál működés közben alkalmazott módon és vezérléssel ellenőrizhetjük. Az ellenőrzés 1,5 percet vesz igénybe (5% idővesztés), és hibátlan végrehajtása után a keresés a korábban megjegyzett ciklusszámnak megfelelő ponttól folytatható. Hiba esetén a fiók tartalékra kapcsolási igényt jelez lekérdezésekor,

és a megelőző sikeres ellenőrzési pontban feljegyzett ciklusszámot közli (ami szintén tárol). Ennek az értéknek megfelelő pontról indul a tartalék fiók keresési tevékenysége. (Részleteket az olvasó Diffie és Hellman idézett [12] cikkében talál.) Érdeemes megjegyezni, hogy még 1%/1000 óra meghibásodási arány mellett is csak 8% annak a valószínűsége, hogy két ellenőrzés között egy adott fiók meghibásodjon.

Természetesen a DES-pártiakat ennek a rejtjelfejtő berendezésnek a megépíthetősége nem ingatta meg, mert még elvi megalkotói is elírmerik, hogy megépítése csak jelentős erőösszpontosítással és csak egy hatalmas szervezetben képzelhető el. Döntőnek tartják azonban azt a tényt, hogy az áramköri technológia fejlődésének eddigi üteméből következtetve 10 éven belül a berendezés egy nagyságrenddel olcsóbban megvalósítható lesz, ami viszont egyenértékű a DES teljes hatástalanításával.

5. Nyilvános kulcsú rejtés

A klasszikus rejtjelezési eljárások mindegyikében nemcsak a fejtésnek, hanem a rejtésnek is van titkosan kezelendő részlete. Ez annak a következménye, hogy a rejtés teljes ismeretében a fejtés minden részletére gazdaságosan következtetni tudunk. Más szóval a rejtés gazdaságosan invertálható függvény. A nyilvános kulcsú rejtjelezési eljárások ezt a gazdaságos invertálhatóságot szüntetik meg, és ezzel lehetővé teszik, hogy az E rejtést minden részletében nyilvánosságra hozzuk, mert ismeretében a titkosan kezelt D megfejtésre gazdaságosan nem tudunk következtetni. Formálisabban, egy (E,D) függvénpár pontosan akkor nyilvános kulcsú rejtjelező eljárás, ha

- a) $D(E(M)) = M$, azaz D az E inverze;
- b) egy összetartozó (E,D) pár előállítása könnyű számítási feladat;
- c) E ismerete nem szolgáltat hatékony módszert D meghatározásához.

Az ezeknek a feltételeknek eleget tevő E függvényeket egyirányú függvényeknek nevezzük. (Az angolszász irodalomban a szemléletes „csapóajtó” — *trap-door* — jelzővel is illetik őket.) Egy nyilvános kulcsú rejtjelezési eljárás a digitális kézjegy megvalósítására is „elegáns” módszer, ha a pótlólagos

- d) $E(D(M)) = M$

megfordíthatósági feltételt is kielégíti. Mivel ebben az esetben E értelmezési tartománya és értékkészlete azonos, E -t egyirányú permutációnak nevezzük. A digitális kézjegyet eredményező eljárással azonban csak a későbbiekben foglalkozunk.

A nyilvános kulcsú rejtjelezés gondolatát elsőként Diffie és Hellman [31] vetették fel, de konkrét eljárást akkor még nem tudtak adni rá. Jelezték azonban, hogy a nyilvános kulcsú eljárások szerkesztésének mindig nehezen megfordítható matematikai feladaton kell alapulnia.

Ebben a fejezetben a jelenleg ismert két legjobban „sikerült” eljárást ismertetjük. Az első, az ún. **MIT-eljárás** [32], azt a tényt hasznosítja, hogy az egész számok törzstényezőkre bontása lényegesen nehezebb feladat, mint összeszorzásuk. Nevét onnan kapta, hogy az MIT (Massachusetts Institute of Technology) egyetem három diákja, R.L. Rivest, A. Shamir, L. Adleman, dolgozta ki. (Kidolgozóikról szokták még RSA-eljárásnak is nevezni.) A másik, a **Merkle–Hellman módszer** [35] az ún. **lefedési feladat** megoldásának nehezségére alapoz. Ez a feladat a következő: ismert a lefedő szakaszok egy halmaza és adott egy lefedendő szakasz. Lefedhető-e ez utóbbi a lefedő szakaszok valamilyen kombinációjával, ha bármelyik lefedő szakaszt csak egyszer használhatjuk fel? És ha igen, melyekkel?

5.1 Az MIT-eljárás

Az MIT-rendszert alkalmazó felhasználó a rejtés és a fejtés kulcsaként egyaránt egy-egy számpárt — (e,n) , illetve (d,n) — használ, amelyek közül az elsőt nyilvánosságra hozza, a másikat — pontosabban abból a d -t — titkosan kezeli. Mind a rejtés, mind a fejtés mod- n hatványozással történik. Rejtéshez ennek megfelelően a teljes nyílt szöveget olyan blokkokra tördeljük, amelyek numerikus értéke egyenként nem haladja meg $(n-1)$ -t. Ha tehát $0 \leq M < n$, akkor az M üzenet rejtett megfelelője:

$$C = E(M) = M^e \pmod{n}.$$

Hasonlóképpen a megfejtés

$$M = D(C) = C^d \pmod{n}.$$

(Vegyük észre, hogy a rejtés nem növeli az üzenet méretét. Mind a nyílt, mind a rejtett üzenet numerikus értéke 0 és $n-1$ közé eső egész szám). A kulcsokban szereplő három egész szám — e, d és n — meghatározási folyamata a következő:

Először választunk két igen nagy (pl. egyenként 100 jegyű) prímszámot, p -t és q -t. Ezek szorzata lesz n :

$$n = p * q .$$

Bár n -et rejtőkulcs részeként nyilvánosságra hozzuk, p és q gyakorlatilag rejtve marad annak következtében, hogy (a kb. 200 jegyű) n törzstényezőkre bontása a gyakorlati lehetőségeket meghaladó nehézségű feladat. Ugyanez akadályozza meg azt is, hogy d -t az e -ből hatékony módszerrel meghatározhassuk. A megfejtés kulcsát, d -t, úgy választjuk ki véletlenszerűen egy igen nagy számtartományból, hogy $(p-1)(q-1)$ -hez relatív prím legyen, azaz

$$\text{Inko } (d, (p-1)(q-1)) = 1 .$$

Végül e a d multiplikatív inverze mod $(p-1)(q-1)$:

$$e * d = 1 \pmod{(p-1)(q-1)} .$$

Ezzel a rejtjelezési eljárás minden összetevője a kezünkben van. Bizonyítanunk kell azonban, hogy E és D egymásnak valóban inverzei; hatékony algoritmust kell adnunk a rejtés és a fejtés kivitelezésére, valamint p , q és e meghatározására; és végül vizsgálnunk kell a védelmi képesség mértékét is.

A rejtés és a fejtés mindkét sorrendben alkalmazva ugyanazt az eredményt adja:

$$D(E(M)) = (M^e)^d = M^{ed} \pmod{n}$$

és

$$D(E(M)) = (M^d)^e = M^{ed} \pmod{n} .$$

Bizonyítanunk tehát az

$$M^{ed} = M \pmod{n},$$

illetve ed definíciója miatt az

$$M^{k(p-1)(q-1)+1} = M \pmod{n}$$

egyenlőséget kell.

Induljunk ki a „kis” Fermat-tételből (p prímszám)

$$a^{p-1} = 1 \pmod{p} \quad 0 < a \leq p-1$$

Ha M és p relatív prímek, akkor $M=cp+r$, $0 \leq r < p$, és $M^{p-1}=(cp+r)^{p-1}$ kifejtésének tagjai r^{p-1} kivételével mind p többszörösei, tehát

$$M^{p-1} = r^{p-1} = 1 \pmod{p}$$

vagy $k(q-1)$ -edik hatványra emelve

$$M^{k(p-1)(q-1)} = 1 \pmod{p}.$$

Végül M -mel szorozva

$$M^{k(p-1)(q-1)+1} = M \pmod{p}.$$

Ez azonban akkor is teljesül, ha M többszöröse p -nek (ilyenkor mindkét oldal értéke $0 \pmod{p}$), tehát **minden** M -re igaz. Hasonló gondolatmenettel

$$M^{k(p-1)(q-1)+1} = M \pmod{q}.$$

Mivel p és q prímszámok, ebből a két egyenlőtlenségből már következik

$$M^{k(p-1)(q-1)+1} = M \pmod{pq} = M \pmod{n}.$$

Ugyanis $a=b \pmod{m_1}$ -ből és $a=b \pmod{m_2}$ -ből $a-b=k_1m_1$ és $a-b=k_2m_2$ következik. Ha azonban m_1 és m_2 relatív prímek, $a-b$ az m_1m_2 többszöröse is kell, hogy legyen. Tehát $a-b=0 \pmod{m_1m_2}$ vagy $a=b \pmod{m_1m_2}$.] Az E és D transzformáció tehát valóban egymás inverze. E és D alkalmazási sorrendjének felcserélhetősége miatt az MIT-eljárás egyirányú permutáció (kielégíti a d) feltételt), és mint ilyen a jelenleg ismert egyetlen.

A rejtés és a fejtés első pillanatra igen költséges műveletnek látszik, mert e és d igen nagy szám lehet. Előrelátó szervezéssel azonban a rejtés (fejtés) legfeljebb $2\log_2 e$ szorzás és $2\log_2 e$ osztás végrehajtását igényli. Írjuk fel e -t bináris alakban:

$$e = \epsilon_k 2^k + \epsilon_{k-1} 2^{k-1} + \dots + \epsilon_1 2 + \epsilon_0.$$

Ezzel

$$M^e = (M^{\epsilon_k})^{2^k} (M^{\epsilon_{k-1}})^{2^{k-1}} \dots (M^{\epsilon_1})^2 M^{\epsilon_0} = \\ = ((\dots (M^{\epsilon_k})^2) M^{\epsilon_{k-1}})^2 \dots M^{\epsilon_1})^2 M^{\epsilon_0}.$$

Tehát

1. Legyen $\epsilon_k \epsilon_{k-1} \dots \epsilon_1 \epsilon_0$ az e bináris ábrázolása.
2. $C \leftarrow 1; i \leftarrow k;$
3. $C \leftarrow C^2 \pmod{n};$
ha $\epsilon_i = 1$, akkor $C \leftarrow CM \pmod{n}$
4. $i \leftarrow i-1$, ha i nem negatív ismétlés 3.-tól.
5. Vége.

Ennek a módszernek a neve „**hatványozás ismételt négyzetre emeléssel és szorzással**”, és sebessége kb. a fele a legjobb ismert eljárásokénak. Mivel e és d egyaránt 200 jegyű lehet, bináris kifejtésük kb. 660 bitnyi. Ennek megfelelően 1320 szorzás és 1320 osztás jelenti a lehetséges legnagyobb munkaigényt. Ez modern számítógépen, 200 jegyű üzenetet feltételezve is, néhány másodpercnyi idő alatt elvégezhető; speciális berendezéssel pedig lényegesen gyorsabban is kivitelezhető.

Egy felhasználó a saját (E, D) párjának meghatározását a két prímszám — p és q — kiválasztásával kezdi. Mindkét prímszámot igen nagyra kell azonban választanunk, hogy $n = pq$ törzstényezőkre bontása gyakorlatilag lehetetlenné váljék. (Emlékeztetőül: p és q nem kerül nyilvánosságra, a felhasználó titokban tartja és gondosan őrzi őket). Egyaránt 100 jegyű prímszámok 200 jegyű n -et eredményeznek, ami már hatásosan megakadályozza a tényezőkre bontást.

A 100 jegyű prímszámok előállításához 100 jegyű (páratlan) véletlenszámokat állítunk elő mindaddig, amíg egy prímszámot nem kapunk. A prímszám-tétel szerint ez kb. $(\ln 10^{100})/2 \cong 115$ kísérlet után következik be. Egy szám prímszám voltának kiderítésére sokféle módszer áll rendelkezésünkre. Az egyik legegyszerűbb és legelegánsabb **Solovay és Strassen** [32] nevéhez fűződik.

Egy nagy u szám vizsgálatához egyenletes eloszlás szerint választunk másik v véletlenszámot az $[1, u-1]$ intervallumból, és megnézzük, hogy

$\text{Inko}(u, v) = 1$ és $J(v, u) = v^{(u-1)/2} \pmod{u}$ — ahol $J(a, b)$ az ún. **Jacobi-szimbólum** — együtt teljesül-e. Ha u prímszám, akkor mindkettő teljesül, ha u összetett, akkor valamelyik nem teljesülésének legalább $1/2$ a valószínűsége. Ha azonban 100 függetlenül választott v mindegyikére együtt teljesülnek, akkor legfeljebb 2^{-100} a valószínűsége, hogy u összetett szám. (Ez az eset is ellenőrizhető néhány rejtési-fejtési kísérlettel, mert összetett modulussal a fejtés nem a kívánt eredményt adja.)

Amikor u páratlan, $v \leq u$ és $\text{Inko}(u, v) = 1$, a Jacobi-szimbólum $[-1, 1]$ lehet, és hatékonyan kiszámítható az alábbi módon:

$$J(a, b) = \text{if } a=1 \text{ then } 1 \text{ else} \\ \text{if } a \text{ páros then } J(a/2, b) (-1)^{(b^2-1)/8} \\ \text{else } J(b \pmod{a}, a) (-1)^{(a-1)(b-1)/4}$$

Azért, hogy a „mindenre figyelő”, tényezőkre bontó algoritmusokkal szemben is

védekezzünk, p és q hossza néhány jegyben különbözzék egymástól, $p-1$ és $q-1$ egyaránt tartalmazzon nagy prímtényezőt, és legnagyobb közös osztójuk lehetőleg kicsi legyen. A középső feltétel teljesítéséhez például a p -t így választjuk: keresünk egy u prímszámot a már elmondottak szerint, majd az $i \cdot u + 1$, $i=2,4,6 \dots$ sorozatot képezzük; p ebben a sorozatban az első prímszám lesz. Egy modern számítógépen a prímszámvizsgálat néhány másodpercet, az egy adott pontot követő első prímszám megkeresése pedig egy-két percet vesz igénybe.

Az e és d paraméter meghatározása már egyszerű. A megfejtés kulcsát, d -t egy nagy számnak választjuk, hogy kimerítéses kísérletezéssel ne lehessen meghatározni. Az euklideszi algoritmussal ellenőrizhetjük, hogy d és $(p-1)(q-1)$ relatív prímek-e, és ekkor meghatározhatjuk e -t is mint d multiplikatív inverzét mod $(p-1)(q-1)$. Ha e kisebbre adódna, mint $2 \log_2 n$, akkor válasszunk új d -t. Ez biztosítja, hogy — a triviális $M=0$ vagy $M=1$ üzenet kivételével — minden rejtett üzenet keresztülmenjen mod- n redukáláson.

Vajon mit mondhatunk ennek a rejtjelezési eljárásnak a védelmi képességeiről? Az mindenesetre nyilvánvaló, hogy ha a behatoló n -et tényezőkre tudja bontani, célját máris elérte, mert akkor e -ből d -re következtethet. Bár a tényezőkre bontás nem bizonyíthatóan nehéz feladat, a mai ismert leghatékonyabb eljárásnak [32] egy 200 jegyű n felbontásához közelítőleg $3,8 \times 10^9$ évre van szüksége, ha egy-egy elemi művelet (100–200 jegyű számok összeadása, kivonása, szorzása, osztása) időigénye $1 \mu s$. Ez az érték jelentős biztonsági tartalékot nyújt a jövő technológiai fejlődésével szemben is. Fennáll ugyanakkor annak a lehetősége, hogy a tényezőkre bontás területén drámai hatékonyságnövekedés következik be. Ezzel szemben áll az a tény, hogy a **törzstényezőkre bontás jól ismert feladat**, és az utóbbi 300 évben a legjobb matematikusok dolgoztak minél hatékonyabb megoldásán. Ez feljogosít bennünket arra, hogy kijelentsük: az MIT-módszer az n felbontásával kísérletező behatolási kísérletek gyakorlati próbáját már kiállta.

Természetesen a felbontásán kívül elképzelhetők egyéb behatolási ötletek is. (Például $(p-1)(q-1)$ közvetlen meghatározása n -ből annak alapján, hogy éppen $\varphi(n)$ -nel egyezik meg. Az ún. **Euler-féle $\varphi(n)$ függvény** az n -nél kisebb és hozzá relatív prím számát adja meg.) Az eljárás kidolgozói ezeket a lehetőségeket is sorra vették, és megmutatták, hogy ezek bármelyikének sikere új törzstényezőkre bontási eljárást is eredményezne. Nehézségük tehát azonos a tényezőkre bontásával.

5.2 A Merkle – Hellman módszer

A matematika egyik régóta jól ismert nehéz feladata az ún. **lefedési feladat**. Ennek egydimenziós változata a következő. Adott a lefedő szakaszok egy $A = \{a_1, a_2, \dots, a_n\}$ halmaza és a lefedendő S szakasz. Lefedhető-e pontosan S az A elemeinek valamilyen részhalmazával? Másképpen fogalmazva: keresendő, ha egyáltalán létezik, egy olyan n -dimenziós bináris x vektor,

hogy $S = x a$ ahol $a = (a_1 a_2 \dots a_n)$ a lefedő szakaszok hosszaiából alkotott vektor.

(Az angolszász irodalomban ezt az iszák — angolul **knapsack** — problémának is nevezik, mert ugyanezen feladattal kerül szembe az a menetelésre induló katona, aki iszákját a lehető legjobban meg akarja tölteni a különféle méretű élelmiszerkonzervekkel). Egy feltételezett x megoldás helyessége könnyen ellenőrizhető, és legfeljebb n összeadás kell hozzá.

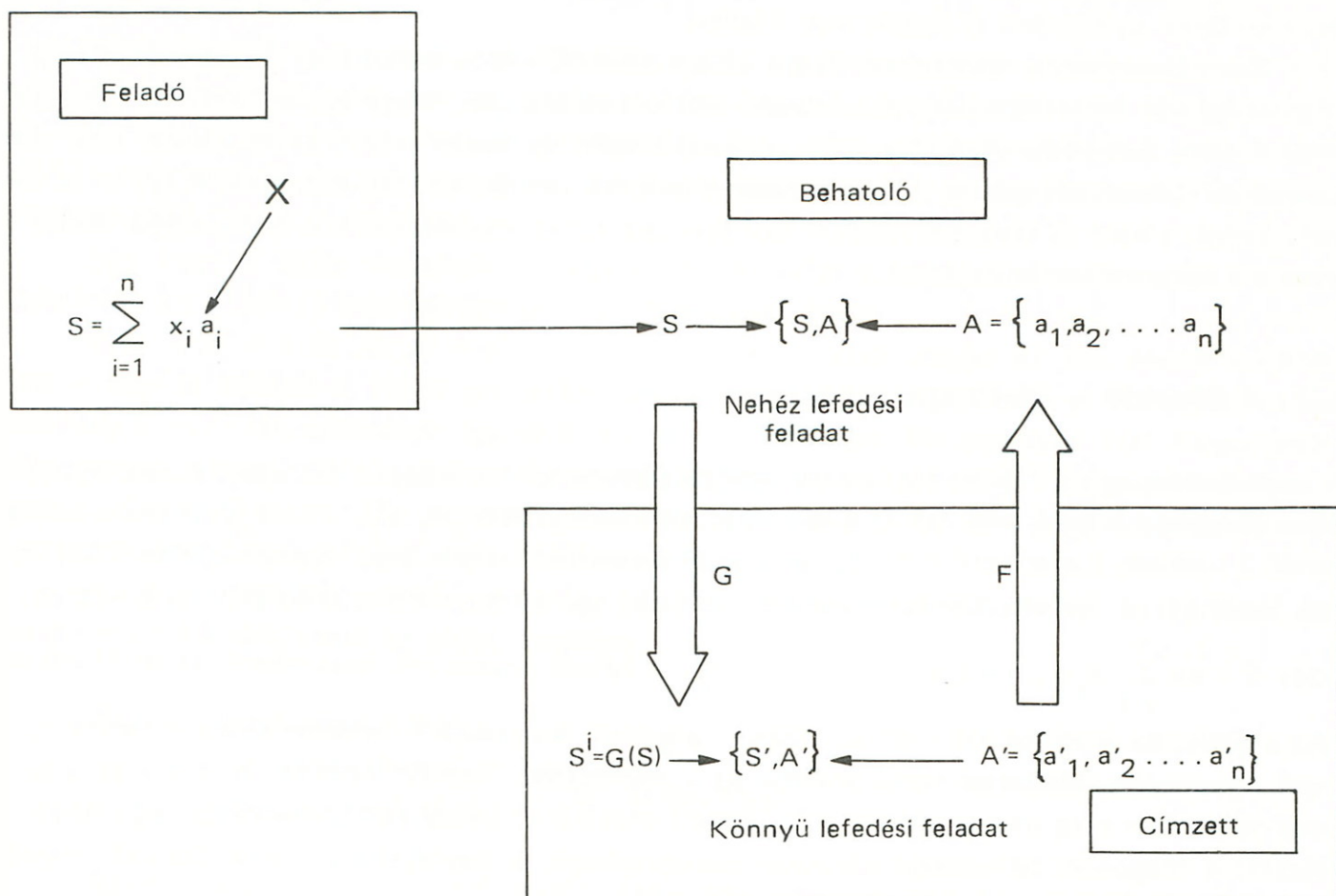
Egy megoldás előállításához a mai ismereteink szerint azonban általában az n -nel exponenciálisan növvő számú műveletre van szükség. A 2^n darab lehetséges x lefedő vektor kimerítéses kipróbálása kivihetetlenné válik, ha értéke meghaladja a 100–200-at. A legjobb, ez ideig napvilágot látott megoldás idő- és tárolókapacitás-igénye egyaránt $2^{n/2}$ -nel arányos.

Mivel a lefedési feladat nehéz, felhasználható információ rejtésére. Ha a nyílt információt n bites blokkokra bontjuk, akkor egy n -elemű $A = \{a_1, a_2, \dots, a_n\}$ lefedő halmaz segítségével egy tetszőleges – vektorként kezelt – n bites x blokk értékét az $S = xa =$

$= \sum_{i=1}^n x_i a_i$ összegbe rejthetjük. Egy külső szemlélő ebből az eredeti x blokkra egyszerű módszerrel nem tud következtetni. A rejtés „szépséghibája” csak az, hogy ha S -et x rejtett megfelelőjének tekintjük, és felhasználásával x -et védetten továbbítjuk, a címzett sem lesz képes x -et hatásosan visszanyerni. Ha pedig ez mégis sikerülne, mi a biztosíték arra, hogy adott S és A mellett csak egyetlen megoldás létezik?

A lefedési feladat nehézsége azonban igen erősen függ a lefedő halmaz megválasztásától. Ha például $A = \{1, 2, 4, \dots, 2^{n-1}\}$, akkor $S = \sum_{i=1}^n x_i 2^i$ és x meghatározása S -ből egyenértékű S bináris ábrázolásának előállításával, ami köztudottan egyértelmű és egyszerű feladat. Ugyanakkor, ha A elemeit egy adott tartományból egyenletes eloszlással véletlenszerűen válogatjuk, elegendően nagy n esetén nagy valószínűséggel nehéz lefedési feladatot állítunk elő. Ezek a megfigyelések egy olyan **egyirányú lefedési feladat** megalkotásának alapötletét sugallják, amely nyilvános kulcsú rejtjelezési eljárásként használható. A lényegét a 5.1 ábra mutatja.

A rejtjelezési eljárás megalkotója egy **könnyű $\{S', A'\}$ lefedési feladathoz** tartozó



5.1 ábra.

$A' = \{ a'_1, a'_2, \dots, a'_n \}$ lefedő halmazból indul ki. Ennek elemeiből egy titkosan kezelt F leképezéssel előállítja az $A = \{ a_1, a_2, \dots, a_n \}$ lefedő halmazt, amely már nehéz $\{ S, A \}$ lefedési feladatot eredményez, és ezt nyilvánosságra hozza. A leképezést úgy választja meg, hogy az A' -be épített belső szerkezet — amely az eredeti lefedési feladat egyszerű megoldhatóságának feltétele — A -ból már se látható, se kikövetkeztethető ne legyen. Az eljárás megalkotója az F függvény megválasztásakor egyúttal egy másik G függvény birtokosává is válik, s így az A -val képzett bármely nehéz lefedési feladatot átalakíthatja egy A' -vel képzett könnyű lefedési feladattá. Pontosabban az

$S = \sum_{i=1}^n x_i a_i$ összeget egy olyan $S' = G(S)$ összegbe transzformálja, amely ugyanakkor az x -nek felel meg, de az eredeti könnyű lefedési feladat lefedő halmazának elemeiből áll: $S' = \sum_{i=1}^n x_i a'_i$. Az $\{ S, A \}$ lefedési feladat megoldása helyett tehát az $\{ S', A' \}$ lefedési feladatot oldja meg.

A rejtjelezett kapcsolat a következőképpen működik. A nyilvánosságra hozott A halmaz tulajdonosának címzett n bites x adatblokkokat az A elemeivel $S = \sum_{i=1}^n x_i a_i$ módon rejtjük, és S -et továbbítjuk. Mivel A nehéz lefedési feladatot határoz meg, S -ből x -re következtetni gyakorlatilag lehetetlen. A címzett, az A halmaz tulajdonosa, viszont először előállítja $S' = G(S)$ -t, majd a titkosan kezelt A' lefedő halmaz birtokában a könnyű $\{ S', A' \}$ lefedési feladatot megoldva jut hozzá az x üzenethez.

A könnyű lefedési feladat bázisául A' elemeit egy szupernövekvő sorozatnak választjuk,

$$a_i > \sum_{j=1}^{i-1} a_j$$

(Vagyis a sorozat minden eleme nagyobb, mint a megelőző összes elem összege! A korábban említett 2 hatványait tartalmazó lefedő halmaz, $A' = \{ 1, 2, \dots, 2^{n-1} \}$, ennek speciális esete.) Egy szupernövekvő sorozattal meghatározott lefedési feladatot a következőképpen oldhatunk meg. Megkeressük azt a legnagyobb indexű a_{i_1} elemet, amely még nem nagyobb S -nél: $a_{i_1} \leq S < a_{i_1+1}$. Az i_1 -nél nagyobb indexű elemek nyilvánvalóan nem jöhetnek szóba S képzésében, a_{i_1} -nek viszont részt kell benne vennie, mert a nála kisebb indexű elemek együttesen is kisebb értéket képviselnek. Ezután az $S^{(1)} = S - a_{i_1}$ különbséggel mindezt megismételjük. A teljes eljárást addig ismételjük, amíg az utolsó különbséget már egyetlen elem le nem fedi (ez következik be, ha S -et az a_{i_j} -k összegeként határoztuk meg), vagy az utolsó különbség már a legkisebb lefedő elemnél is kisebb (ez fordulhat elő, ha S -et véletlenszerűen választottuk).

Az A' lefedő halmaz transzformációjához két nagy egész számot választunk, m -et és w -t, amelyek egymáshoz képest relatív primek. Ez utóbbi biztosítja egy olyan w^{-1} létezését, amely w -nek inverze mod m :

$$w w^{-1} = 1 \pmod{m}.$$

Az A' elemeiből

$$a_i = w a'_i \pmod{m}$$

formában állítjuk elő A elemeit. A szorzás-maradékképzés ismert statisztikai tulajdonsága miatt az a_i -k álvéletlenszámok lesznek, tehát a $[0, m-1]$ tartományban egyenletesen oszlanak meg. Az m -et és w -t nem ismerő behatoló számára $\{S, A\}$ tehát nehéz lefedési feladatot jelent. Az eljárás tervezője azonban S helyett az

$$S' = w^{-1} S \pmod{m}$$

összeggel foglalkozik. Mivel

$$S' = w^{-1} S = w^{-1} \sum_{i=1}^n x_i a_i = w^{-1} \sum_{i=1}^n x_i w a'_i = \sum_{i=1}^n x_i a'_i \pmod{m}$$

ha m -et úgy választjuk meg, hogy

$$m > \sum_{i=1}^n a'_i$$

akkor

$$S' = \sum_{i=1}^n x_i a'_i.$$

Az x nyílt blokk tehát az $\{S', A'\}$ könnyű lefedési feladat megoldásával meghatározható.

Ha $n=100$ -at választunk, ami biztonsági megfontolásokból az alkalmazható blokk-hossznak körülbelül az alsó határa, az egyes paramétereket az alábbi tartományokból választhatjuk egyenletes eloszlással:

$$a'_i \in [(2^{i-1} - 1)2^{100} + 1, 2^{i-1} 2^{100}]$$

$$m \in [2^{201} + 1, 2^{202} - 1],$$

$$w' \in [2, m-2] \text{ és } w = \frac{w'}{\text{Inko}(w', m)}$$

Ezzel mind a szuper-növekedést, mind az m -re kirótt alsó értékhatár feltételt teljesítjük, és egyben minden lefedő elem megválasztásához egy 2^{100} nagyságú tartományt biztosítunk. Egy behatoló tehát még egyetlen lefedő elem értéktartományát sem tudja próbálgatással kimeríteni. A transzformált lefedő elemek értéke 1 és $m-1$ közé esik, tehát 202 bites ábrázolást kívánnak. Mivel S legfeljebb 100 ilyen elem összege és $\log_2 100 \sim 7$ ($2^7 = 128$), legfeljebb 209 bit hosszúságú. Ez a rejtjelezés tehát redundáns, és a rejtés 2.09:1 arányú **blokkméret-növekedéssel** jár.

Szupernövekvő sorozat alkalmazása miatt az A' lefedő halmaz legkisebb és legnagyobb eleme nagymértékben eltér egymástól. Ez a tény némi támpontot adhat a rejtjelfejtőnek [36]. Ezt a gyengeséget kiküszöbölhetjük, ha az eredeti lefedő halmazban strukturált számokat használunk, amelyek alakja:

$$\begin{array}{l} a'_1 = \text{---} \text{---} \text{---} \text{---} \text{---} 0 \dots \dots \dots 0 \text{---} \text{---} \text{---} \text{---} \text{---} \\ a'_2 = \text{---} \text{---} \text{---} \text{---} \text{---} 0 \dots \dots \dots 0 \text{---} \text{---} \text{---} \text{---} \text{---} \\ a'_n = \left(\begin{array}{c} \text{véletlen} \\ \text{bitsorozatok} \end{array} \right) \text{---} \text{---} \text{---} \text{---} \text{---} 0 \dots \dots \dots \left(\begin{array}{c} \text{szupernövekvő} \\ \text{sorozat} \end{array} \right) \text{---} \text{---} \text{---} \text{---} \text{---} \end{array}$$

Az eredeti lefedő halmaz szerkezeti tulajdonságainak eltakarására a másik hatásos módszer a (w, m) transzformáció ismételt alkalmazása. Az eredeti a'_i lefedő elemeket sorban több különböző $(w_1, m_1), (w_2, m_2), \dots, (w_k, m_k)$ számpárral transzformáljuk

$$\begin{aligned} a_i^{(1)} &= w_1 a'_i \pmod{m_1} \\ a_i^{(2)} &= w_2 a_i^{(1)} \pmod{m_2} \\ &\vdots \\ a_i^{(k-1)} &= w_{k-1} a_i^{(k-2)} \pmod{m_{k-1}} \\ a_i^{(k)} &= w_k a_i^{(k-1)} \pmod{m_k} \end{aligned}$$

$$\begin{aligned} S^{(k-1)} &= w_k^{-1} S \pmod{m_k} \\ S^{(k-2)} &= w_{k-1}^{-1} S^{(k-1)} \pmod{m_{k-1}} \\ &\vdots \\ S' &= w_1^{-1} S^{(1)} \pmod{m_1} \end{aligned}$$

Többszöri (w, m) transzformáció alkalmazásával azt is megengedhetjük magunknak, hogy az eredeti lefedési feladatot egészen speciálisan válasszuk meg:

[illegible]

61

goltuk”, az üzenet S' -nek a diagonális mátrixhoz tartozó helyértékeiről közvetlenül leolvasható.

Az ismételt transzformációk mindegyikének megválasztásakor be kell tartani azt a követelményt, hogy n nagyobb legyen, mint a pillanatnyi a_i -k összege. Ha $n=100$, ez azt jelenti, hogy m bináris ábrázolása legalább 7 bittel lesz nagyobb. Ebből következik, hogy egy transzformációs lépés után az a_i -k (legfeljebb) 7 bittel hosszabak, mint elődjeik. Minden iterációs lépés tehát az a_i -k hosszát (legfeljebb) 7 bittel növeli. Ha például 20 iterációs lépést alkalmazunk, és az a_i -k hosszúsága kezdetben 200 bit, akkor az eredő lefedő elemek hossza 340 bit, és S 347 biten ábrázolható (mert 100 a_i összege lehet). Rejtéskor tehát a blokkhossznövekedés 3,47:1 arányú.

A matematikának a lefedési feladat is jól ismert és sokszor vizsgált problémája. Megoldására speciális feltételek mellett hatékony és gyors módszerek születtek. A (w, m) transzformációval (transzformációkkal) előállított lefedési feladatra ezek a módszerek nem alkalmazhatók, a behatoló csak a megoldás általános módszereire támaszkodhat. Ezekre érdemes egy pillantást vetnünk, mert általánosítható gondolatokat tartalmaznak.

A rejtjelfejtés egyik szélsőséges módszere a kimerítésez kísérletezés. A lefedő halmaz elemeiből alkotható összegeket valamilyen szisztematikus sorrendben előállítjuk, és minden egyes lépésben összehasonlítjuk őket az adott S -sel. A másik véglet az lehet, hogy a lefedő halmaz ismeretében egyszer határozzuk meg a lehetséges összegeket, és (pl. nagyság szerint) rendezett formában tároljuk őket. Ekkor a rejtjelfejtés (pl. bináris) kereséssel valószínűsíthető meg. Az első változat tárigénye minimális, legrosszabb esetben azonban 2^n számú kísérletet kíván; a második viszont csakély időigényű (pl. a bináris keresés n lépésben célhoz vezet), de 2^n különböző összeg (és a hozzá tartozó x érték) tárolását igényli. Találhatunk azonban „középutas” megoldást is, amelyhez $2^{n/2}$ lépésre és $2^{n/2}$ elemű tárhelyre van szükség. Az eredeti lefedő halmazt két részhalmazzá bontjuk, amelyek mindegyike $n/2$ elemet tartalmaz. Az egyik részhalmaz elemeiből képezhető összegeket előre elkészítjük, és egy $2^{n/2}$ elemű táblában nagyság szerinti sorrendben tároljuk. Egy rejtjelfejtési feladat során a másik részhalmaz elemeiből képezhető összegeket rendszerezetten állítjuk elő. Ha egy lépésben ez az összeg Σ , akkor $S - \Sigma$ -t megkereséljük az előre elkészített táblában megkeresni. A megfejtést akkor kapjuk meg, ha ez a keresés sikeres. A keresett x megoldás egyik felét a lépés során kipróbált kombináció, a másik felét pedig a táblázatban fellelt kiegészítő összegnek megfelelő kombináció adja. Ez a módszer nyilvánvalóan legfeljebb $2^{n/2}$ kísérlettel sikerre vezet. A „tér-idő kompromisszum” tehát jelentősen javítja a behatolás esélyeit. Ha azonban $n=100$, akkor egyaránt 2^{50} lépésszám és 2^{50} elemű tárhely-igény adódik, ami továbbra is kizárja a keresés sikerét.

Az eljárás bemutatására szolgáló kis méretű példához legyen $A = \{24, 57, 13, 75, 5, 30\}$ és $S = 104$. A lefedő halmazt kétfelé bontjuk: $\{24, 57, 13\}$ és $\{75, 5, 30\}$. Az első részhalmazból képezhető összegeket és a hozzájuk tartozó fél értékeit táblázatosan tároljuk:

0	13	24	37	57	70	81	94
000	001	100	101	010	011	110	111

Ezután az x maradék feléhez (x') tartozó összegeket rendszerezett formában előállítjuk, és kiegészítőjükkal ($s - x'$) minden lépésben az előző táblázathoz fordulunk.

<i>lépés:</i>	1	2	3	4	5	6	7
x'	000	001	010	011	100	101	110
<i>összeg:</i>	0	30	5	35	75	105	80
kiegészítő:	104	74	99	69	29	-1	24

A keresés itt a 7. lépéssel ér véget, mert csak a 24 eleme az előre elkészített táblának. A teljes megfejtés első fele a táblázatból 100, a második fele a sikeres keresési lépésből 110.

6. Kulcsgondozás

A rejtjelezés rendszerteknikai feladatainak körvonalazásakor (2.2 fejezet) már említettük, hogy a védett kapcsolat kulcsa a behatoló közvetlen támadásának természetes célpontja. Egyrészt a sikeres behatolás útja megkerüli a rejtjelfejtés fáradságos és legtöbbször sikertelen kísérleteit, másrészt egy kulcs élete folyamán sok és többféle behatolási alkalmat kínál fel. A 2.3 ábra szerint a rejtjelező kulcsok biztonsága érdekében háromféle védelmi feladatot kell megoldanunk. Lehetetlenné kell tennünk, hogy a behatoló a kulcselőállító rendszert birtokba vegye; gondoskodnunk kell a kulcsok biztonságos tárolásáról; és a behatolótól védve kell eljuttatnunk egy kapcsolat kulcsát a felhasználóihoz. Ezek a feladatok többnyire egymástól függetlenül megoldhatók, és mindegyikre többféle megoldást is ismerünk. Egy rendszerben azonban a három feladat kölcsönhatásba lép, és a választott megoldásoknak harmonizálniuk kell. Mivel a kulcstovábbításra kiválasztott eljárás meghatározó szerepet játszik, elsőként a **kulcskiosztás** témakörével foglalkozunk. Ezután vesszük sorra a **kulcselőállítás** és a **kulcstárolás** módszereit.

6.1 Kulcskiosztás

A kulcsok célba juttatásának alapproblémáját az jelenti, hogy legalább a kapcsolatba lépő partnerek egyikéhez a kulcsot ugyanazon a közegen át kell eljuttatni, amelyet éppen a kulcs felhasználásával kívánunk védetté tenni. Mivel a kulcs védelem nélküli továbbítása az egész védelmi rendszert hatástalanítja, a kulcsok továbbítására védett és az adatkapcsolatok rendszerétől független megoldást kell találnunk.

6.1.1 Merkle „rejtvény” módszere

A legkevesebb előfeltételt támasztó és legkevesebb előkészületet kívánó kulcstovábbító módszert **R. Merkle** alkotta meg [41]. A módszer azonban — éppen azért, mert a védett csatornát szinte „a semmiből akarja elővarázsolni” — nem nyújtja a gyakorlat megkövetelte védelmet; megnehezíti a behatoló feladatát, de a behatolást nem teszi (gyakorlatilag) lehetetlenné. Röviden mégis ismertetjük, mert alapgondolata szellemes, és általánosíthatósága nincs kizárva.

A kapcsolatot kezdeményező felhasználó véletlenszerűen előállít N kulcsot, amelyek közül partnere fogja kiválasztani a kapcsolat tényleges kulcsát. Mindegyik K_i kulcshoz hozzárendel egy ugyancsak véletlenszerűen kiválasztott, de egyértelmű azonosítót. A ka-

pott $[ID_i, K_i]$ párok mindegyikét más-más kulccsal rejtve $M_i = E(K_i, [ID_i, K_i])$ formában elküldi őket partneréhez. A K_i kulcsokat azonban úgy választja meg, hogy egy-egy M_i -t elfogadható időn belül meg lehessen fejteni. A DES alkalmazása esetén például az 56 bites kulcs felső 25–35 helyértékét megállapodás szerint zérusnak választja. A partner tehát csak egy $2^{20}–2^{30}$ méretű kulcstartománnyal áll szemben, ha egy rejtett kulcsot és az azonosítóját meg akarja ismerni. Egy ilyen méretű keresési feladatot pedig meg is tud oldani. A partner a vett rejtett üzenetek közül kiválaszt egyet és megfejti. A benne talált K_j kulcsot a kapcsolat kulcsának tekinti, az ID_j azonosítót pedig nyílt formában visszaküldi. Ennek vétele után a kezdeményező fél is ismeri a kapcsolat közös K_j kulcsát.

Ezalatt természetesen a behatoló is jelen van, tehát minden M_j üzenet és ID_j is birtokába kerül. Az ID_j -hez tartozó kulcsot azonban csak akkor ismeri meg, amikor az ID_j -t tartalmazó üzenetet megfejti. Ehhez azonban átlagosan csak $N/2$ kísérlettel jut el, feladata tehát $N/2$ -szer nagyobb munkaigényű, mint a címzett partneré. Ez a munkai-gény-növekedés azonban nem zárja ki hatékonyan a behatolót, mert a kapcsolatfelvétel során átküldhető adatmennyiséget gyakorlati megkötések korlátozzák.

6.1.2 Nyilvános kulcsú kiosztás

A kulcskiosztásnak ez a módja a következő. A számítógépes rendszerben működik egy nyilvános kulcsú rejtjelező rendszer, amelynek minden felhasználó tagja. (Mindegyiküknek van kihelyezett, mindenki számára elérhető rejtőkulcsa.) A rendszer egyidejűleg egy konvencionális rejtjelező rendszer használatát is támogatja. A nyílt kapcsolat felvétele után a kezdeményező fél választ egy konvencionális rendszerben használható kulcsot, majd ezt a nyilvános kulcsú rendszer segítségével választott partneréhez védetten eljuttatja. Ezt követően a kapcsolat tartása a konvencionális rendszer segítségével és az átküldött kulcs felhasználásával történik.

Az ellentmondás nyilvánvaló: ha egyszer van egy nyilvános kulcsú rendszer, miért nem azt használjuk a teljes kapcsolat tartama alatt? Az ismertetett stratégia azonban észszerű megfontolásokon nyugszik. A konvencionális algoritmusok alkalmazása lényegesen gazdaságosabb: egyrészt egyszerűbbek és gyorsabbak, mert egyszerűbb műveletekből épülnek fel, másrészt a DES jóvoltából ma már egyetlen áramkörbe építve hozzáférhetőek. Ezenkívül váratlan meglepetéstől is kevésbé kell tartanunk használatuk esetén. Ha azonban teljes mértékben a konvencionális rendszerre hagyatkoznánk, a kulcs védettsége még akkor is a rendszer „diszkréciójától” függ, ha azt mi állítottuk elő. (A kulcstovábbító csatorna védettségéről ugyanis csak a rendszer tud gondoskodni.) A nyilvános kulcsú rendszer megbízható működéséhez lényegesen kevesebb és könnyebben megvalósítható feltétel szükséges. (Lásd a Digitális kézjegy és a Partnerazonosítás c. fejezetet.) Ugyanakkor a védett kapcsolat felvétele közben a nyilvános kulcsú eljárások rosszabb hatásfoka nem döntő tényező.

Az előző fejezetben ismertetett bármelyik, nyilvános kulcsú rejtjelezési eljárást felhasználhatjuk a kapcsolat felvétele során. Most egy további nyilvános kulcsú eljárást mutatunk be. Részben, mert egy újabb, nehezen megfordítható matematikai műveletet alkalmaz, részben pedig azért, mert csak kulcskiosztásra alkalmas.

Az eljárás alapja az, hogy véges testekben a hatványozás egyszerű művelet, míg mai ismereteink szerint megfordítása, a **(diszkrét) logaritmusképzés** munkaigényes ($p^{1/2}$ nagyságrendű, ahol – a prímszám – p a test elemeinek száma). A módszer a következő.

Két felhasználó U_i és U_j mindegyike választ egy X_i , illetve X_j egész számot, és ezekkel előállítja az

$$Y_i = \alpha^{X_i} \pmod{p}$$

illetve

$$Y_j = \alpha^{X_j} \pmod{p}$$

mennyiségeket, ahol a p prímszám és α (a p elemű véges test — $GF(p)$ — egy primitív eleme) nyilvánosan ismert érték. Ezt követően Y_i -t és Y_j -t kicserélik, majd megállapodás-szerűen a

$$K_{ij} = \alpha^{X_i X_j} \pmod{p}$$

kulcsot (vagy ha K_{ij} mérete nem megfelelő, annak valamilyen közös elhatározott függvényét) használják konvencionális rejtjelezésű kapcsolatuk létrehozására. Ezt bármelyik partner a

$$K_{ij} = Y_i^{X_j} = Y_j^{X_i} \pmod{p}$$

összefüggés alapján viszonylag egyszerűen elő tudja állítani. Egy behatoló azonban csak a nyilvánosságra hozott Y_i és Y_j értéket ismeri, így K_{ij} -t csak a

$$K_{ij} = Y_i^{\log_{\alpha} Y_j} = Y_j^{\log_{\alpha} Y_i} \pmod{p}$$

összefüggés alapján határozhatja meg, amihez a logaritmusképzést fel kell használnia. (Vegyük észre, hogy — mint minden nyilvános kulcsú rendszerben — most is meg kell akadályoznunk a behatolót abban, hogy belépjen az Y -ok cseréjének útjába. Ellenkező esetben mindkét partnert megtévesztheti saját Y -jának behelyettesítésével.)

Ennek a kulcstovábbító rendszernek a védettsége a (diszkrét) logaritmusképzés munkaigényétől függ. **Hellman és Pohling** kimutatta [34], hogy az általuk kidolgozott és a korábbiaknál hatékonyabb logaritmusképzési algoritmus akkor nem jelent javítást az ismert ($p^{1/2}$ nagyságrendű műveletszámot igénylő) módszerekkel szemben, ha $p-1$ törzstényezőinek legnagyobbika azonos nagyságrendű p -vel. Optimális választást tehát $p-1=2p_r$ eredményezne, ahol p_r is prímszám. (Lásd a MIT-eljárás prímszám kiválasztási feladatát!) Mivel $GF(p)$ elemei $\log_2 p$ bites számok, a mod- p hatványozás $\log_2 p$ nagyságrendű elemi művelettel és ugyanezzel arányos méretű tártartományban hajtható végre. A p prímszámot elegendően nagyra választva a rejtjelezési és behatolási munka ráfordítási aránya tehát a kívánt mértékűre állítható be.

6.1.3 Alapkulcsrendszer

A konvencionális rejtjelezési eljárást alkalmazó rendszerekben a kulcsok biztonságos szétosztásához szükséges védett adatátviteli csatornákat a **mesterkulcsok** segítségével kialakított **alapkulcsrendszer** adja. Az alapkulcsrendszerben minden egyes rendszerelemnek (számító-

gép, terminál stb.) saját rejtjelező kulcsa, ún. **mesterkulcsa** van, amelyet a szóban forgó berendezésben nyílt formában, de fokozott mértékben védett helyen (a rejtjelező berendezésben) tárolunk. Az egyes rendszerelemekben **minden más kulcs** (tehát még a többi elem mesterkulcsai is.) saját **mesterkulccsal rejtjelezett formában** és az átlagosan védett tárolók valemelyikén található meg. Az így kialakuló alapkulcsrendszerben a mesterkulcsok kezelése (előállításuk, a rendszer elemeibe juttatásuk) a rendszeren kívül történik, és védelmüket is a rejtjelező berendezés **fizikai védelme** nyújtja. A mesterkulcsok rendszerét azért nevezzük alapkulcsrendszernek, mert csak a rendszer működéséhez szükséges kulcsok továbbítására, illetve védelmére szolgál.

A rejtjelezett kulcsok használata azonban új rejtjelezési műveleteket kíván, amelyek a rejtjelezendő adatok mellett nem nyílt, hanem rejtett formájú kulcsot fogadnak el. Jelölje MK_0 egy rendszerbeli gazdagép mesterkulcsát. Ebben a gazdagépben egy tetszőleges **kapcsolatkulcs**, KK csak $E_{MK_0}(KK)$ formában áll rendelkezésünkre. A rejtő és fejtő transzformáció tehát:

$$\begin{aligned} ECPH \{ E_{MK_0}(KK), adat \} &= E_{KK}(adat) \\ DCPH \{ E_{MK_0}(KK), E_{KK}(adat) \} &= adat. \end{aligned}$$

Megvalósításuk természetesen a rejtjelezési alapeljárás E és D transzformációjával történik:

$$\begin{aligned} ECPH \{ K, X \} : KK &= D_{MK_0}(K), Y = E_{KK}(X), \\ DCPH \{ K, X \} : KK &= D_{MK_0}(K), X = D_{KK}(Y). \end{aligned}$$

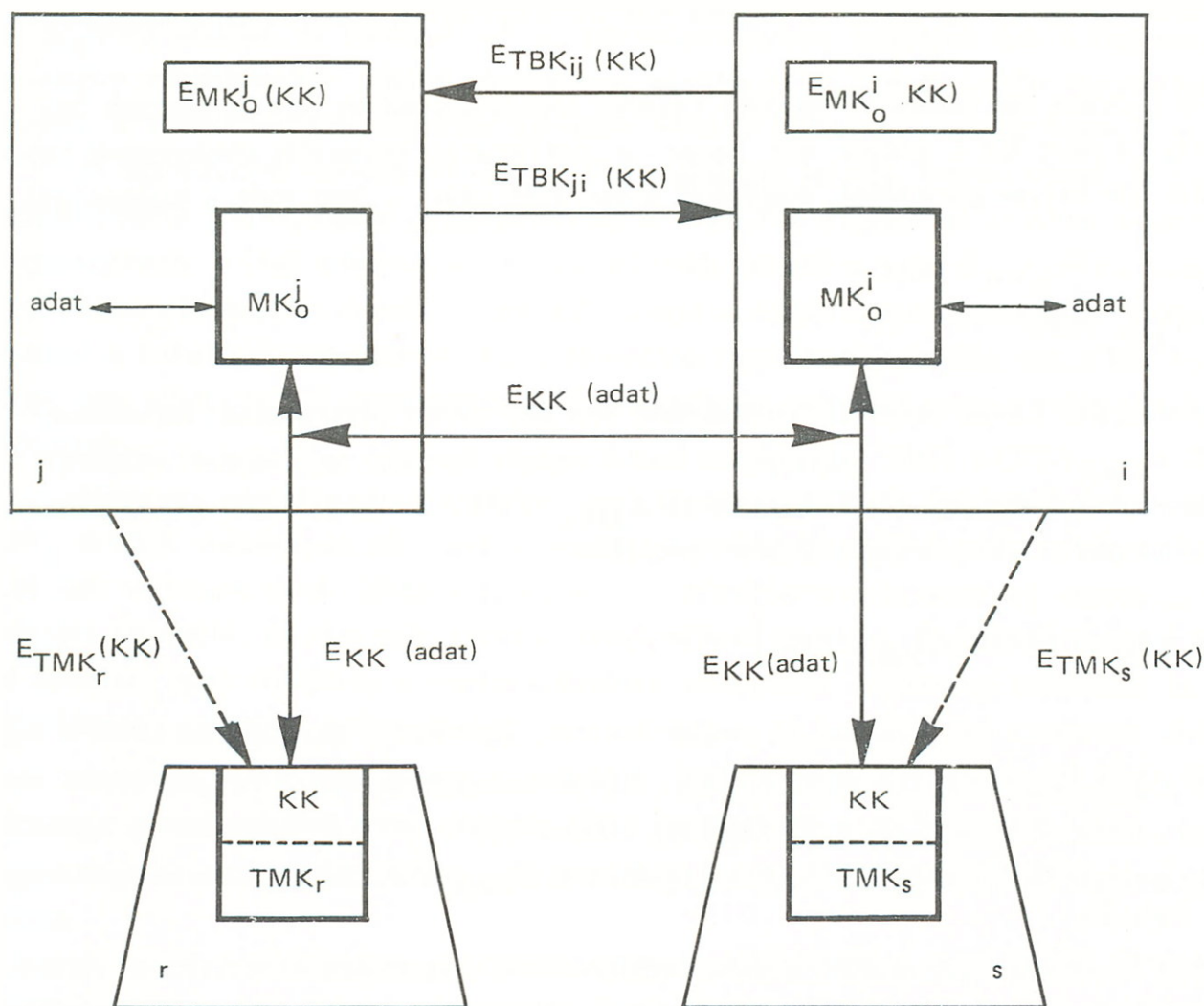
A rejtjelezés lépései azonban hozzáférhetetlenek, mert a jól védett rejtjelező berendezés hajtja őket végre. A mesterkulcs a műveletek egy harmadik, rejtett paramétere. A behatoló ennek következtében az esetleg megszerzett rejtett kulcsot **csak ugyanabban a rendszerben hasznosíthatja**, amelyből megszerezte őket. Ez a védelmi sérüléseket lokalizálja, és a behatolás felfedésének esélyeit jelentősen javítja.

Elsősorban ellenőrzési célokra a rejtjelező berendezés

$$EMK \{ adat \} = E_{MK_0}(adat)$$

műveletével az MK_0 szerinti rejtés is megvalósítható.

A kulcsgondozás további részleteinek kidolgozásához a 6.1 ábrán vázolt rendszerszerkezetet vesszük figyelembe. Ebben az egyes gazdagépek végzik a kulcsok előállítását a hálókörökbe tartozó terminálok igényeinek kielégítésére, és a gazdagépek közötti protokoll teszi lehetővé tetszőleges rejtjelezett kapcsolat felépítését. A gazdagépben egy **terminál mesterkulcsát** rejtjelezett formában tároljuk, egy terminálnak szánt minden kulcsot a terminál mesterkulcsával rejtjelezve továbbítunk. A kapcsolatok rejtjelezését azonban a kapcsolatok felépítésekor dinamikusan előállított és hozzárendelt kulcs, a **kapcsolatkulcs** segítségével végezzük. A rendszer működésének egyetlen pillanatában sem tartózkodhat kulcs nyílt formában a gazdagép operatív tárában.



6.1 ábra.

Az elmondottak értelmében egy tetszőleges kapcsolatkulcsot a gazdagépben $E_{MK_O}(KK)$ alakban tárolunk és használunk fel rejtjelezésre. Ugyanezt a KK kulcsot előállítás után $E_{TMK}(KK)$ formában közöljük a terminállal, amelynek TMK a mesterkulcsa. A terminál rejtjelező berendezése ugyancsak rejtett kulcs fogadására készül fel, ezért az

$$ECPH \{ E_{TMK}(KK), adat \} = E_{KK}(adat) ,$$

$$DCPH \{ E_{TMK}(KK), E_{KK}(adat) \} = adat$$

műveleteket hajtja végre. A terminál azonban az egyszer meghatározott KK kapcsolatkulcsot tárolja, és közvetlenül használja mindaddig, amíg a fenti műveletek első paraméterét elhagyjuk:

$$ECPH \{ , adat \} = E_{KK}(adat) ,$$

$$DCPH \{ , E_{KK}(adat) \} = adat .$$

A gazdagéptől érkező kapcsolatkulcsot a terminál a

$$DMK \{ E_{TMK}(KK) \} = KK$$

művelettel csomagolja ki, de ennek a műveletnek az eredménye kívülről nem hozzáférhető.

A nyílt kulcsok tárolására vonatkozó tilalmat legegyszerűbben úgy tarthatjuk be, hogy közvetlenül nem KK -t állítjuk elő, hanem a véletlenszám-generálás eredményét (minden nyolcadik bit helyes paritásnak megfelelő kiigazítása után — lásd még a kulcselőállítást!)

$$RN = E_{MK_0}(KK)$$

módon értelmezzük. Ekkor azonban szükségünk van egy olyan oszthatatlan műveletre, amely ebből $E_{TMK}(KK)$ -t TMK gazdagépen belüli rejtett formája segítségével előállítja. Eddigi feltételezésünk szerint TMK ugyanis az $E_{MK_0}(TMK)$ formában áll rendelkezésünkre. Vegyük azonban észre, hogy a $DCPH$ megfejtés

$$DCPH \{E_{MK_0}(TMK), E_{TMK}(KK)\} = KK$$

alkalmazásával egyszerre két tilalmat is megsérthetünk. Egyrészt a gazdagépen belül a kapcsolatkulcsot nyílt formában elő lehet állítani, másrészt a gazdagépen kívül jogtalanul megszerzett $E_{TMK}(KK)$ mennyiséget a gazdagépen belül jogosan lehet felhasználni a behatolás céljaira. Az $E_{MK_0}(TMK)$ rejtés tehát nem korlátozza $E_{TMK}(KK)$ alkalmazhatóságát megszerzésének helyére.

Nyilvánvaló tehát, hogy a terminálok mesterkulcsait egy másik kulccsal kell rejtteni. Ezt a gazdagépben a mesterkulcsból és a rejtjelező berendezésben állítjuk elő (például megfelelő bitinverziók és pozíciópermutáció alkalmazásával). Ha ezt az új mesterkulcsváltozatot MK_1 -gyel jelöljük, akkor

$$RFMK \{E_{MK_1}(TMK), E_{MK_0}(KK)\} = E_{TMK}(KK)$$

definícióval új műveletet kell bevezetnünk, hogy a rejtetten előállított és tárolt kapcsolatkulcsból a továbbított formáját előállíthassuk. Az új művelet természetesen ismét az eredeti E és D alkalmazásával valósítható meg:

$$RFMK \{TK, K\} : TMK = D_{MK_1}(TK), KK = D_{MK_0}(K) \rightarrow E_{TMK}(KK) .$$

MK_1 -nek MK_0 -ból való előállítása megengedhető, mert minden elfogadható rejtjelezési eljárás kísérletileg kimutatott tulajdonsága, hogy már egyetlen kulcsbit megváltoztatása is előre meg nem jósolható változásokhoz vezet ugyanannak a nyílt szövegnek a rejtett változatában. Kapcsolt (erősen függő) kulcsok alkalmazása sem vezet tehát a megfelelő rejtett szövegekben olyan összefüggéshez, amely a rejtjelfejtő feladatát megkönnyítené. (Még akkor sem, ha MK_1 képzési szabályát ismerné is!)

Az eddigiekben adatkapcsolatok védelmét vizsgáltuk. Magától adódik a kérdés: használható-e az ismertetett kulcsgondozási stratégia tárolt adatok védelmére is? A válasz egyszerűnek látszik: használjuk egy adatállomány védelméhez az FK rejtjelező kulcs

$E_{MK_0}(FK)$ alakját, és tároljuk azt az adatállományhoz kötötten (pl. a fejében). Az adatállomány előállításakor, illetve olvasásakor ezután az $ECPH$ és $DCPH$ műveleteket alkalmazhatjuk. E közvetlen megoldásnak azonban néhány hátrányos tulajdonsága is van.

A legnagyobb nehézséget $E_{MK_0}(FK)$ védelme jelenti, hiszen ennek birtokában az adatállomány megfejtése a gazdagépen belül a $DCPH$ művelettel minden további nélkül végrehajtható. Külső adathordozón való tárolása lehetővé teszi $E_{MK_0}(FK)$ rendszerbe épített védelmet megkerülő eltulajdonítását, majd a rendszerbeni (most már jogos) alkalmazásával a behatolás befejezését. Ez a lehetőség tehát hatástalanítja azt a törekvésünket, hogy egy adathordozó rendszertől való eltávolítása ne csökkentse a rajta tárolt adatok védettségét.

Mivel az adatállományok élettartama általában hosszú, fel kell készülni arra is, hogy MK_0 értékét valamilyen ok miatt egy adatállomány élettartama alatt meg kell változtatni. Elő kell tudnunk tehát állítani az egyes FK adatállománykulcsok új mesterkulccsal rejtjelezett formáját, és régi változatukat ezzel ki kell cserélni. Ennek bármilyen megvalósítása azonban igen nehézkes a mesterkulcsokra vonatkozó biztonsági előírások és az érintett adatállományok nagy száma miatt.

Végül $E_{MK_0}(KK)$ gazdagéphez kötött, az adatállomány más gazdagépen nem fejthető meg. Ehhez ugyanis MK_0 -t egy másik gazdagép számára hozzáférhetővé kellene tenni, ami azonban a teljes alapkulcsrendszer létjogosultságát kétségbe vonó védettséggyengítés lenne.

A felsorolt nehézségeket elkerülhetjük, ha az adatállomány-rejtjelező kulcsokat egy külön **adatállomány-mesterkulcs**, FMK segítségével rejtjük, és adatállománnyal együtt $E_{FMK}(FK)$ -t tároljuk. (Mivel FMK nem hagyja el a gazdarendszert, az adatállomány védettsége eltávolításával nem csökken, továbbra is FMK védettségével egyezik meg.) FMK -t szintén nem nyílt formában tároljuk, hanem egy MK_2 (az MK_0 -ból MK_1 -hez hasonló módon, de különbözőképpen származtatott) kulccsal rejtjelezve. Az adatállomány védelmét egy új,

$$RTMK\{E_{MK_2}(FMK), E_{FMK}(FK)\} = E_{MK_0}(FK)$$

módon értelmezett művelet teszi lehetővé, mert alkalmazása után az $ECPH$ elrejtés és $DCPH$ megfejtés felhasználhatóvá válik.

Az FMK adatállomány-mesterkulcsot MK_0 -val ugyanazon okok miatt nem rejthetjük, amiért TMK -t sem. Kissé bővebb magyarázatra szorul, hogy miért nem használhatjuk MK_1 -et sem. Tegyük fel, hogy rendelkezésünkre áll egy $RTMK^*$ művelet az $RTMK$ -éhoz hasonló, de MK_2 helyett MK_1 -et hasznosító definícióval. Ezt

$$RTMK^*\{E_{MK_1}(TMK), E_{TMK}(KK)\} = E_{MK_0}(KK)$$

formában alkalmazva elérhetnénk, hogy $E_{TMK}(KK)$ -t továbbításakor a rendszeren kívüli lehallgatással megszerezve később a rendszerben jogosan felhasználhassuk behatolásra. ($RTMK^*$ és $RFMK$ az MK_0 és TMK szempontjából „inverz” művelet.)

Az ismertett kulcsgondozó rendszert sokgépes rendszerre a *TBK* kulcstovábbító kulcsok segítségével terjeszthetjük ki. Egy TBK_{ij} kulcs mind az i , mind a j gazdarendszerben ismert, és az i -edik gazdagéptől a j -edik gazdagép felé irányuló rejtett kulcstovábbításra használjuk. Az ellenkező irányban használt továbbító kulcs TBK_{ji} , és általában különbözik TBK_{ij} -től. Másképpen: a TBK_{ij} kulcs az i -edik gazdagép számára **átadókulcs**, a j -edik gazdagép számára **átvevőkulcs**. Mielőtt az i -edik gazdagépből a j -edik gazdagéphez továbbítanánk, egy adatállomány fejében $E_{FMK_i}(FK)$ -t az

$$\begin{aligned} RTMK\{E_{MK_2^i}(FMK_i), E_{FMK_i}(FK)\} &= E_{MK_0^i}(FK), \\ RFMK\{E_{MK_1^i}(TBK_{ij}), E_{MK_0^i}(FK)\} &= E_{TBK_{ij}}(FK) \end{aligned}$$

műveletek végrehajtása után $E_{TBK_{ij}}(FK)$ -val helyettesítjük. Ebből az átküldés után a gazdagépen az

$$RTMK\{E_{MK_2^j}(TBK_{ij}), E_{TBK_{ij}}(FK)\} = E_{MK_0^j}(FK)$$

átalakítás után olyan formában áll rendelkezésünkre, amelynek segítségével (és *DCPH*-t alkalmazva) az adatállomány tartalma hozzáférhetővé válik. Figyeljünk fel arra, hogy a kulcstovábbítás lebonyolításához egy gazdagépben az átadókulcsokat MK_1 -gyel, az átvevőkulcsokat viszont MK_2 -vel rejtett formában kell tárolni.

Az i -edik gazdagépben előállított kapcsolatkulcs $E_{MK_0^i}(KK)$ alakjából a j -edik gazdarendszerben használható $E_{MK_0^j}(KK)$ alakot hasonlóképpen állíthatjuk elő. Az i -edik gazdagépben KK -t az

$$RFMK\{E_{MK_1^i}(TBK_{ij}), E_{MK_0^i}(KK)\} = E_{TBK_{ij}}(KK)$$

átalakítással készítjük elő továbbításra, a j -edik gazdagépen használhatóvá pedig

$$RTMK\{E_{MK_2^j}(TBK_{ij}), E_{TBK_{ij}}(KK)\} = E_{MK_0^j}(KK)$$

módon tesszük. A különböző gazdagépek termináljai között így védett kapcsolatot tudunk kiépíteni.

A kapcsolat létrehozása közben a két gazdagép között az $E_{TBK_{ij}}(KK)$ mennyiségeket továbbítjuk. A behatoló, ha meg is szerzi ezeket, az i -edik gazdagépben nem tudja hasznosítani, mert TBK_{ij} -t az $RTMK$ művelet alkalmazásához szükséges $E_{MK_2^i}(TBK_{ij})$ formában az i -edik gazdagépen nem tároljuk. A j -edik gazdagépben viszont hasznosíthatja, ezért (és az adatállomány-védelemben betöltött szerepe miatt is) az $RTMK$ művelet alkalmazását csak szigorúan ellenőrzött és szabályozott körülmények között szabad megengedni. Erről és a kulcsok tárolás közbeni védelméről a gazdagépnek az egyéb információk és adatok védelmét ellátó védelmi rendszere gondoskodik.

6.2 Kulcselőállítás

Bármilyen konvencionális rejtjelező rendszerben a legsebezhetőbb láncszem mindig a kulcsok előállítási folyamata, mert a szükséges nagy mennyiségű kulcsot csak rendszeren belüli algoritmikus módszerrel lehet előállítani. Ebből viszont számos behatolási lehetőség adódik.

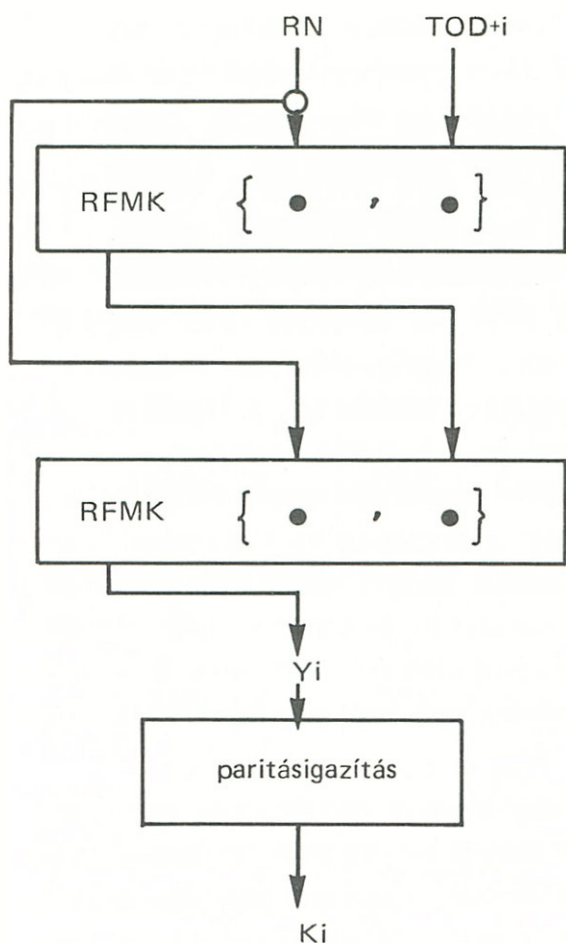
Az előállító folyamat a kulcsokat (vagy azok egy meghatározó részét) általában nyílt formában állítja elő, és csak a kiosztásuk előtt rejtjelezi őket. Ez lehetővé teszi, hogy a behatolás a kulcselőállítás folyamat saját tartományának megfigyelésével (olvasásával) történjen. Ennek megakadályozása a gazdagép belső védelmi rendszerének a feladata.

Behatolhat valaki a végfelhasználók megtévesztésével is: a behatoló elérheti, hogy a rendszer kulcselőállító algoritmusának helyébe lépjen akár a gazdarendszeren belül annak védelmi rendszerét kijátszva, akár az adatközlő hálózatban a gazdagép és a terminál közé beékelődve. A megtévesztett felhasználó ezután a behatolótól kapott kulcs használatának idején értékes információkat és adatokat szolgáltat ki akarátán és tudtán kívül. Ennek a behatolási lehetőségnek a kivédéséhez a kapcsolatba lépni szándékozó partnerek kölcsönös azonosításának megbízható és behatolásbiztos megvalósítására van szükség (lásd Kapcsolathitelesítés).

A kulcselőállítás általában ismételhető algoritmus segítségével történik, az egymást követően előállított kulcsok között ezért meghatározott kapcsolat áll fenn. A teljes védelmi rendszert hatástalaníthatja, ha erre a kapcsolatra a behatoló a kulcsok egy részének ismeretében következtetni tud. (A behatoló ugyanis lehet a rendszer felhasználója, és ezért számos – esetenként egymást követően előállított – kulcshoz törvényesen is hozzájuthat. Mivel a kész kulcsokat a felhasználók több-kevesebb ideig tárolják, az is feltételezhető, hogy egy behatoló néhány idegen kulcshoz is hozzájut. A behatoló számára a teljes védelmi rendszer hatástalanításának költsége tehát a legrosszabb esetben is a néhány előfizetőhöz való behatolás – vagy megvesztegetésük – költségének szintjére csökken).

Ez utóbbi behatolási fenyegetés ellen leghatásosabban úgy védekezhetünk, hogy a kulcsokat a **rendszeren kívül** és az eredmény szempontjából **megismételhetetlen** (tehát valódi és nem álvéletlenszám-előállításon nyugvó) **algoritmussal** állítjuk elő. A hangsúly a megismételhetetlenségen van, mert ez egyben az előállított kulcsok függetlenségét is biztosítja. A rendszeren kívüliség csak az előállított kulcsok védettségét növeli, mert az egyetlen gyenge láncszem – a fogadó és felhasználó rendszerbe ültetés – is szabályozott körülmények között (pl. a rendszer üzemidején kívül) végezhető el. Ez a kulcselőállítási és szétosztási módszer azonban meglehetősen nehézkes, így csak **kisszámú és statikus** (hosszú élettartamú és alapvető szerepű) **kulcs előállításához** célszerű felhasználni. A korábban ismertetett alapkulcsrendszerben a gazdagépek mesterkulcsai éppen ezt a kulcsosztályt képviselik. A gazdagép-mesterkulcsokat például ismételt pénzfeldobással (a fej-írás sorozatokat 1–0 sorozatként értelmezve) lehet előállítani, beültetésük pedig gyári, fizikailag védett környezetben történhet.

A **másik nagy kulcsosztályt** a terminál-mesterkulcsok vagy bármilyen rendszer felhasználóinak saját alapkulcsai képezik. Ezek száma a rendszeren belül potenciálisan nagy, használatuk statikus, és a már előállított (pontosabban használatban levő) kulcsok ismételt kiosztását meg kell akadályozni. A kulcselőállítás ismételhetősége viszont szükséges követelmény a kulcsvesztéssel járó nagy mennyiségű adatvesztés veszélye miatt. Az alapkulcsrendszerben az ilyen kulcsok előállítására az *RFMK* műveletet hasznosíthatjuk a 6.2 ábrán megadott módon.



$TOD =$ rendszeridő
 $i =$ sorszám
 $Y_i = E_a (DMK_0 (E_a (DMK_0 (TOD+1))))$
 $a = DMK_1 (RN)$

6.2 ábra

Az algoritmushoz a következő megjegyzéseket fűzhetjük:

Az algoritmus egy 64 bites, rendszeren kívül előállított állandó RN véletlenszámnak és a kulcs sorszámának függvényeként állítja elő a kulcsokat. A TOD (rendszeridő) értékek nem a véletlenszerűséget fokozzák, hanem ugyanannak az RN értéknek nem szándékos újbóli felhasználásából eredő kulcssorozat-ismétlés lehetőségét zárják ki.

Mivel RN véletlenszám, $DMK_1 (RN)$ is az. (Egy rejtjelező algoritmus nem eredményez a nyílt szövegtől függő egyszerű kapcsolatokat a rejtett blokk bitjei között). Ezért az Y_i -k (vagyis a K_i kulcsok) két független és titkos paraméter, RN (amelyet a kulcselőállító algoritmus üzemeltetője szolgáltat) és DMK_0 (amely a felhasznált gazdagép része) függvénye. Egy behatoló tehát vagy két adat megszerzésére kényszerül, vagy RN birtokában, de az eredetileg felhasznált gazdarendszerben, az $RFMK$ műveletet igénybe véve kell tevékenykednie. Ez utóbbi használata azonban — mint már említettük is — rendeltetésére való tekintettel szigorúan ellenőrzött. A behatoló feladata tehát mindkét esetben elég nehéz.

Mivel a DES algoritmus behatolásbiztos, a K_i -k (sőt az eredeti Y_i -k) ismerete sem veszélyezteti RN és DMK_0 biztonságát; több kulcs ismerete sem fenyeget a kulcselőállító rendszer birtokbavételével.

Az előállított kulcsok felhasználókhöz rendelését például a következő módon valósítjuk meg. Minden, kulccsal rendelkező felhasználó (kulcsának) alfanumerikus nevét egy táblában tartjuk nyilván a hozzá tartozó TOD értékkel együtt. Valahányszor egy új kulcsigénylő jelentkezik, (kulcsának) alfanumerikus nevét ebben a táblában megkeressük. Ha

a keresés sikertelen marad, akkor a táblában új tételt nyitunk (amelyhez a rendszer pillanatnyi *TOD* értékét használjuk fel). A kulcs előállítását mindig a táblaindexszel (*i*) és a bejegyzett *TOD* rendszeridőértékkel végezzük el. Ez a megoldás biztosítja a kiosztott kulcsok egyértelműségét és az elveszett kulcsok ismételt előállíthatóságát. Ugyanakkor a rendszer védettségét nem csökkenti, mert

- a tábla a rendszeren kívül tartható és így jól védhető, és
- az *RN* érték rendszeren kívüli titkos kezelése még a tábla megszerzése után is meg-
hiúsítja a kulcsok ismételt előállítását.

Egy védett rendszer működéséhez szükséges **harmadik kulcstípust** a kapcsolat-, illetve adatállomány-védő kulcsok képezik. Ezekre elsősorban dinamizmusuk jellemző. A rendszer működése során gyakran és nagy számban van rájuk szükség, tehát a rendszerben állandóan elérhető számítógépes algoritmussal kell előállítanunk őket. A nyílt formájú kulcsok tárolásának tilalmát figyelembe véve az előállított álvéletlenszám nem közvetlenül a kapcsolat- (*KK*) vagy adatállomány-védő (*FK*) kulcs, hanem annak rejtett formája:

$$RN = E_{MK_0} (KK)$$

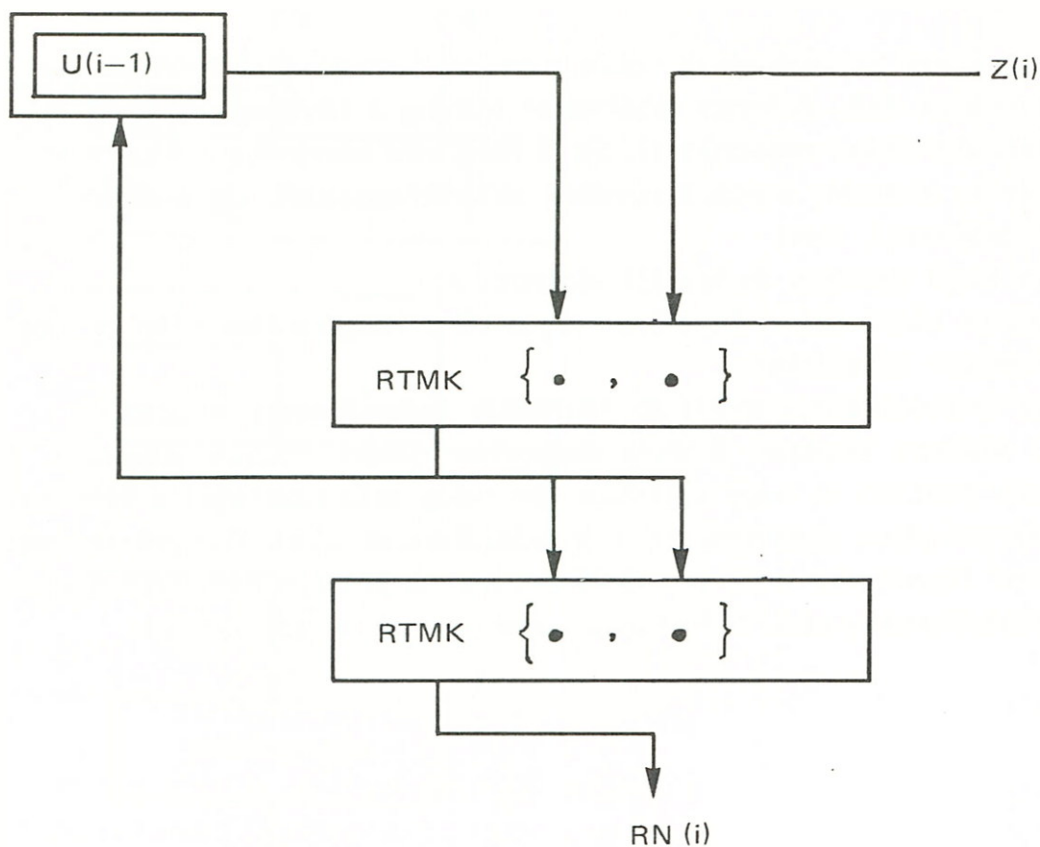
vagy

$$RN = E_{MK_0} (FK) .$$

(A pontosság kedvéért megemlítjük, hogy az így meghatározott *KK*, illetve *FK* a DES alkalmazásakor általában még nem kulcs. Csak úgy válnak azzá, ha minden nyolcadik bitjüket a kulcsoktetten páratlan paritásának megfelelően kiigazítjuk. Ezért a rejtjelezett kulcsparaméterek megfejtését követően az *ECPH*, *DCPH*, *RFMK* és *RTMK* műveletek megvalósításába is be kell iktatni ezt a paritáskiigazítást.)

Kapcsolat- és adatállomány-védő kulcsokat előállító álvéletlenszám-generátort mutat a 6.3 ábra, amely a rendszer *RTMK* műveletét használja. Ennek egyik bemeneti paraméterét a rendszerből származó független $Z_{(i)}$ értékek adják. (Ezek függetlenségét például úgy érhetjük el, ha a rendszerműködés egyes eseményeinek bekövetkezésekor — pl. megszakítások kezelése — leolvasott pillanatnyi *TOD* értéket használjuk fel képzésükre.) A másik bemenő paramétert a korábbi *RTMK* művelet eredményének ($U(i-1)$) visszacsatolásával állítjuk elő. Ennek az értéknek a védelméről úgy gondoskodunk, hogy védett tártartományban tároljuk. Végül $RN(i)$ -t $U(i)$ -ből egy másik *RTMK* művelettel állítjuk elő.

A mesterkulcsokat előállításuk után elhelyezzük a megfelelő rendszerelem rejtjelező berendezésében. Ennek során (és ezt követően) biztonsági okokból a beültetett kulcs kiolvasására többé már nincs lehetőség. Látszólag tehát a beültetés közben elkövetett esetleges hibát nem tudjuk felfedezni. A beültetés eredményét azonban közvetett módon úgy ellenőrizhetjük, hogy a vizsgált egységgel egy ismert eredményű rejtjelezési műveletet hajtunk végre. A gazdagép-mesterkulcsokat például egyszerűen ellenőrizhetjük, ha választunk egy *RN* véletlenszámot és azt a berendezésen kívül MK_0 -val rejtjük: $E_{MK_0} (RN)$. Ezt követően *RN*-re az ellenőrzendő rejtjelező egység *EMK* műveletét alkalmazzuk. Mivel $EMK(RN) = E_{MK_0} (RN)$, a két különböző úton kapott eredmény egyezése közvetetten mutatja, hogy a berendezés valóban az MK_0 kulcsot tartalmazza. Ez az ellenőrzés azonban nem zárja ki annak lehetőségét, hogy mindkét esetben a kívánt mesterkulcs helyett más (de azonos) kulcsot adtunk meg. Ennek az eseménynek a valószínűségét úgy csök-



$U(0)$ = tetszőleges kezdőérték
 $Z(i)$ = két vagy több TOD érték függvénye
 $RN(i)$ = i -edik álvéletlenszám

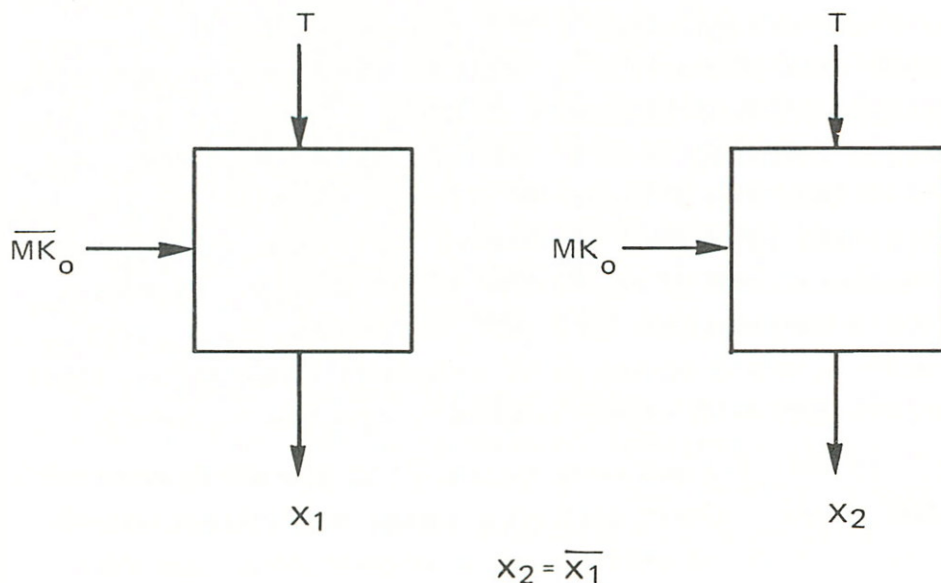
6.3 ábra.

kenthetjük, hogy a két kísérletben nem azonos, de kapcsolt kulcsokat használunk. Erre például a DES szimmetriatulajdonsága nyújt lehetőséget (6.4 ábra). Elsőként MK_0 inverzét, $\overline{MK_0}$ -t helyezzük el a rejtjelező berendezésben, majd egy tetszőleges T értékkel elvégezzük az

$$EMK(T) = E_{MK_0}(T) = X_1$$

rejtést. Ezután betöltjük MK_0 -t, és most T inverzét, $\overline{T}$ -t rejtjük

$$EMK(\overline{T}) = E_{MK_0}(\overline{T}) = X_2.$$



6.4 ábra.

Mivel a második kísérletben mind a kulcs, mind a nyílt adatblokk inverzét használtuk, a DES már említett szimmetriatulajdonsága miatt MK_0 akkor került helyesen a berendezésbe, ha

$$X_2 = \bar{X}_1 .$$

Egy terminál mesterkulcsát ellenőrizhetjük önállóan vagy gazdagépének segítségével. Az önálló ellenőrzéshez egy H_M ellenőrző értékre van szükség, amelyet a terminál-mesterkulcs előállításával egyidejűleg (TID a terminál azonosítója) az

$$\begin{aligned} EMK\{ TMK \} &= E_{MK_0}(TMK), \\ DCPH\{ E_{MK_0}(TMK), TID \} &= D_{TMK}(TID), \\ EMK\{ D_{TMK}(TID) \} &= E_{MK_0}(D_{TMK}(TID)) , \\ H_M = ECPH\{ E_{MK_0}(D_{TMK}(TID), TID \} &= E_{D_{TMK}(TID)}(TID) \end{aligned}$$

műveletsorral állítunk elő. A TMK terminál-mesterkulcs helyessége H_M birtokában önállóan is ellenőrizhető a terminál $ECPH$ műveletének alkalmazásával:

$$ECPH\{ TID, TID \} = E_{D_{TMK}(TID)}(TID) = H_E .$$

Aki tehát TMK -t a terminálba ülteti, a művelet helyességéről H_E és H_M egyenlőségével győződhet meg.

A gazdagép TMK -t közvetett módon mindig ellenőrzi: ha a terminál-mesterkulcs megsérül, a gazdagép és a terminál nem „érti meg” egymást. Egyszerű nyugtázással ez a helyzet ki is deríthető. Miután egy gazdagép elküldött egy új KK kapcsolatkulcsot a terminálhoz, a KK -val rejtve utána küld egy U értéket is, amelynek egy $\phi(U)$ függvényét várja válaszul. A terminál megfejti $E_{KK}(U)$ -t, kiszámítja $\phi(U)$ -t és KK -val rejtve visszaküldi. Ha a terminál mesterkulcsa sértetlen, akkor $\phi(U)$ -t a gazdagép ebből visszanyeri. Ellenkező esetben a terminál már KK -t „félreértene”, és ezt a gazdagép a párbeszéd végén felismeri.

6.3 Kulcstárolás

Az adatok tárolására rejtjelezést használunk, a kulcsok védelmére azonban lényegileg különböző módszert kell találnunk, mert újabb rejtjelezések csak elodázzák, de nem oldják meg a problémát. (Az alapkulcsrendszerben például a mesterkulcsokat már nem tudjuk rejtjelezéssel védeni.) A legbiztonságosabb kulcsgondozás a kulcsot egyetlen, jól őrzött helyen (számítógép, emberi agy vagy páncélszekrény) tárolja. Ez a megoldás azonban nagyon sérülékeny, mert egyetlen balszerencsés esemény (számítógép-meghibásodás, hirtelen halál vagy szabotázs) a kulcsot és egyúttal a vele rejtett adatokat mindörökre hozzáférhetetlenné teheti. Ennek orvoslásaként tárolhatjuk ugyanazt a kulcsot egyidejűleg több különböző helyen is, ez azonban a behatolás veszélyét (számítógép-védelmi sérülés, árusítás, gondatlanság stb.) jelentősen növeli. Ha a különböző helyeken nem a teljes kulcsot, hanem csak egy-egy (önmagában használhatatlan) részletét tároljuk, a védelmet visszaállítjuk, egyidejűleg azonban ismét megjelenik a sérülékenység. Ez a gondolat mégis használható, de

a kulcselőállítás védelmének fokozására. A rendszerben egyidejűleg több kulcselőállító központot üzemeltetünk, egy védett kapcsolatban pedig nem egyedi kulcsokat használunk, hanem több kulcselőállító helyről származó kulcsok valamilyen függvényeként előálló eredő kulcsot. A kapcsolat megnyitásakor például mindegyik kulcselőállító központtól kérünk egy-egy kulcsot, majd ezek mod-2 összegét használjuk kapcsolatkulcsnak.

Tartós kulcsok védelméhez azonban olyan módszerre van szükség, amely a részeken tárolás előnyeinek megtartása mellett jelentősen csökkenti a sérülékenységet is. Ezt az igényünket egy ún. (k, n) küszöbrendszer elégíti ki.

1. A védendő D kulcs előállításához szükséges adatokat n darab egymástól függetlenül tárolt (birtokolt) $D_1, D_2, \dots, D_n$ kulcsrészletbe építjük be.

2. Legalább k kulcsrészlet egyidejű felhasználásával a D kulcs egyszerűen előállítható.

3. Legfeljebb $k-1$ kulcsrészlet ismeretében a kulcs teljes mértékben meghatározatlan marad (abban az értelemben, hogy minden lehetséges értéke továbbra is egyformán valószínű).

A véges geometriák tulajdonságaira alapozott küszöbrendszert **Blakely** [52] dolgozott ki, **A. Shamir** (az MIT nyilvános kulcsú rejtjelezési eljárás „társszerzője”) pedig a polinommal közelítés ismert tulajdonságait használta fel ugyanerre a célra [51]. Ez utóbbi módszer nemcsak elegáns, de igen egyszerű és Blakely módszerénél hatékonyabb is.

A függvényközelítések elméletében közismert eredmény: pontosan egy olyan $(k-1)$ -ed fokú $y=g(x)$ polinom létezik, amely k darab adott $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$ ponton keresztül fektethető (vagyis minden $1 \leq i \leq k$ -ra $y_i = g(x_i)$). Ez az eredmény természetesen igaz marad abban az esetben is, ha a polinom együtthatói nem valós számok, hanem egy véges test — $GF(p)$ — elemei. A 0 és $p-1$ közé eső értékűnek feltételezett D kulcsot ezután így oszthatjuk szét kulcsrészletekre, illetve állíthatjuk azokból vissza.

Együtthatóinak $GF(p)$ -ből való sorsolásával egy $(k-1)$ -ed fokú

$g(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x + a_0$ polinomot választunk, amelynek állandó tagja megegyezik D -vel: $a_0 = D$. Ezután az $x=1, 2, \dots, n$ pontokban kiszámítjuk $g(x)$ értékét:

$$D_1 = g(1), D_2 = g(2), \dots, D_n = g(n).$$

A kulcsrészletek az (i, D_i) párok lesznek.

Bármely k darab kulcsrészlet ismeretében $g(x)$ együtthatóit interpolációval meghatározhatjuk, majd $D=g(0)$ -t a polinom ismeretében egyszerű behelyettesítéssel kapjuk.

A részkulcsok közül bármelyik $(k-1)$ darab ismerete viszont D -ről semmi információt nem közöl. A behatoló ugyanis $k-1$ részkulcs ismeretében minden lehetséges D' -höz ($0 \leq D' < p$) pontosan egy olyan $(k-1)$ -ed fokú $g'(x)$ polinomot szerkeszthet, amelynek az origóban $D'=g'(0)$ az értéke és az ismert pontokon is keresztülmegy. Mivel a ténylegesen felhasznált polinomot véletlen sorsolással választottuk ki, a behatoló által előállított bármelyik polinom azonos valószínűséggel lehet az igazi. A behatolónak tehát D valódi értékére vonatkozóan semmilyen támpontja sincs. Az $n = 2k-1$ választással a (k, n) küszöb módszer igen megbízható, és jól védett kulcs tárolást eredményez. A tárolt kulcsot még $\lfloor n/2 \rfloor = k-1$ kulcsrészlet teljes megsemmisülése esetén is vissza tudjuk állítani a megmaradt kulcsrészletekből, de a behatoló még akkor sem tudja megszerezni, ha a megmaradt k kulcsrészletből bármelyik $\lfloor n/2 \rfloor = k-1$ darabot (egy kivétellel az összeset!) meg is szerzi.

Az eljárás kivitelezéséhez nyilvánvalóan szükséges, hogy P egyaránt nagyobb legyen n -nél és D -nél is. Az utóbbi feltétel azonban mindig teljesíthető azzal, hogy D -t olyan részekre bontjuk, amelyek egyenként mind kisebbek p -nél. Egy egyszerű példa segíthet a megértésben. Ha p -t 2^{16} -nál nagyobbra, de hozzá közel választjuk, 65536 kulcsrészlet használatát tesszük lehetővé. Ez általában bőségesen elegendő, és a számításokhoz csak 16 bites aritmetikát igényel. Egy 64 bites kulcsot viszont 4, egyenként 16 bites részre bonthatunk. Ezeket a részeket külön-külön bontjuk kulcsrészletekre, vagyis egy-egy kulcsrészlet $(i, D_i^{(1)}, D_i^{(2)}, D_i^{(3)}, D_i^{(4)})$ alakú. Visszaállításkor a kulcs egyes részeit külön-külön állítjuk vissza, majd a visszaállított részeket egyetlen kulcsba egyesítjük.

Egy polinom értékének kiszámítására a

$$g(x) = a_{k-1}x^{k-1} + \dots + a_1x + a_0 = ((\dots((a_{k-1}x + a_{k-2})x + a_{k-3})\dots)x + a_1)x + a_0$$

azonosság hatásos módszert ad, az interpoláció elvégzésére pedig — mivel k általában kis egész szám (értéke ritkán haladja meg a 4–5-öt) — a kézenfekvő, lineáris egyenletrendszer-megoldáson alapuló megoldások is megfelelnek.

A (k, n) küszöb módszer néhány előnyös tulajdonsága az egyéb, kombinatorikus jellegű megoldásokkal szemben:

1. A kulcsrészletek mérete nem haladja meg az eredeti kulcsét.
2. Ha k -t rögzítetten tartjuk, kulcsrészleteket dinamikusan állíthatunk elő (ha a rendszerhez új felhatalmazott személy vagy rendszerelem csatlakozik), vagy törölhetünk (ami tulajdonképpen kulcsrészlet-megsemmisülésnek felel meg).
3. Az eredeti kulcs érintetlenül hagyása mellett is könnyű a kulcsrészletek változtatása. Mindössze egy új polinomot kell választanunk, amelynek ugyanaz az állandó tagja. A kulcsrészletek rendszerének gyakori változtatása jelentősen növeli a védettséget, mert behatoláshoz csak azonos „kiadású” (közös polinomból származó) kulcsrészletek használhatók fel.
4. Felhatalmazáskor különböző számú kulcsrészlet kiadásával hierarchikus kulcsörző rendszert valósíthatunk meg. Ha például egy cég elnökének 3, elnökhelyetteseinek 2–2, igazgatóinak pedig 1–1 kulcsrészletet adunk, akkor egy $(3, n)$ küszöb módszeren alapuló rendszerben egy csekket digitálisan az elnök egyedül, egy elnökhelyettes egy másik felelős személlyel együtt vagy három igazgató együttesen írhat alá.

7. Kapcsolathitelesítés

Az előző két fejezetben ismertetett rejtjelezési és kulcsgondozási módszerek alkalmazásával védtelen közegen keresztül is **bizalmas kapcsolat**ot építhetünk fel. Bármely teljes védelmi rendszernek ez az egyik alappillére, mert az illetéktelen behatolót segítségével kizárhatjuk az adatkapcsolaton áramló (illetve adathordozókon tárolt) információ birtoklásából, megértéséből. A másik két alappillért a számítógépen belüli **hozzáférés-védelem** és a **felhasználók** megbízható **azonosítási rendszere** alkotja. Egy bizalmas kapcsolat mit sem ér azonban:

- a) ha benne nem a partnerek szándékainak megfelelő információcsere folyik;
- b) ha a kapcsolat felvételekor a behatoló megtévesztéssel az egyik partner helyébe lép; és végül
- c) a kapcsolaton áramló információval felelősségre vonás veszélye nélkül vissza lehet élni.

a) Egy behatoló a kapcsolaton áramló adatok megértése (megfejtése) nélkül is jelentős kárt okozhat azzal, ha az adatfolyamot a bizalmas kapcsolatot létrehozó partnerek számára leleplezhetetlenül megváltoztatja. Az ilyen **szabotázs** tevékenységnek rejtjelezett **adatok kivonása** (a címzetthez való eljutásának megakadályozása), illetve **visszajátszás** (korábban lehallgatott és tárolt rejtjelezett adatok beiktatása) az alapformái. Szabotázst a rejtjelezéssel **nem tudunk megakadályozni**, célunk csupán a **szabotázs**kísérlet **felismerése lehet**. Ennek módszereit összefoglaló néven **üzenethitelesítés**nek nevezzük.

b) Minden bizalmas kapcsolat felépítése során fennáll annak a veszélye, hogy a behatoló valamelyik partner helyébe lép. Ennek egyik formája kivédhetetlen. Ha a megismerés teljes egészében a számítógépes rendszeren kívül történik — pl.: egy felhasználót azonosító titkos adatok megszerzésével —, leleplezése a rendszerben lehetetlen. Végrehajtását csupán megnehezíteni lehet; ennek módszereivel a **felhasználóazonosítás**nál, egy későbbi fejezetben foglalkozunk. A **partnerazonosítás** módszerei azok ellen a megismerési kísérletek ellen nyújtanak védelmet, amelyek végrehajtásához a behatolónak be kell iktatnia magát a rendszer és a kapcsolatot kereső partnerek közé.

c) Az üzenethitelesítés és partnerazonosítás (a bizalmas kapcsolattal) együttesen csak a **kapcsolat tartamára** és csupán a **partnerek számára** biztosítják a kapcsolaton áramló információ hitelességét. Ha az adatok, információk eredetét egy **harmadik személynek** és a *kapcsolat megszűnte után is* hitelt érdemlően bizonyítani akarjuk, akkor a mindennapi életben ilyen célokra kiterjedten használt *aláírás* megfelelőjére, a **digitális kézjegyre** van szükségünk.

Az üzenethitelesítés, a partnerazonosítás és a digitális kézjegy a **kapcsolathitelesítés** egy-egy témakörét képviseli.

7.1 Üzenethitelesítés

Blokkrejtés alkalmazásakor egy rejtjelezett kapcsolatnak nemcsak az üzeneteit, hanem még az egy üzeneten belüli blokkokat is egymástól teljesen függetlenül rejtjük el. Ez módot adhat arra, hogy a behatoló az adatáramlást észrevétlenül megváltoztathassa a kapcsolatban. A behatoló egyedi blokkokat vagy teljes üzeneteket vonhat ki egy működő kapcsolatból és juttathat vissza a rendszerbe egy alkalmas későbbi időpontban. Ezzel zavart okozhat vagy illetéktelenül információhoz, szolgáltatásokhoz juthat anélkül, hogy a felhasznált rejtjelezett adatok megfejtésével, megértésével foglalkozna! A behatolás ilyen formájához a lehetőséget az teremti meg, hogy egy védett **kapcsolat időbeli folytonosságának**, illetve az **üzenetek időbeliségének** ellenőrzését nem oldottuk meg. A megoldás keresése során két közelálló, de mégis különböző feladattal kerülünk szembe. Megállapíthatóvá kell tennünk, hogy

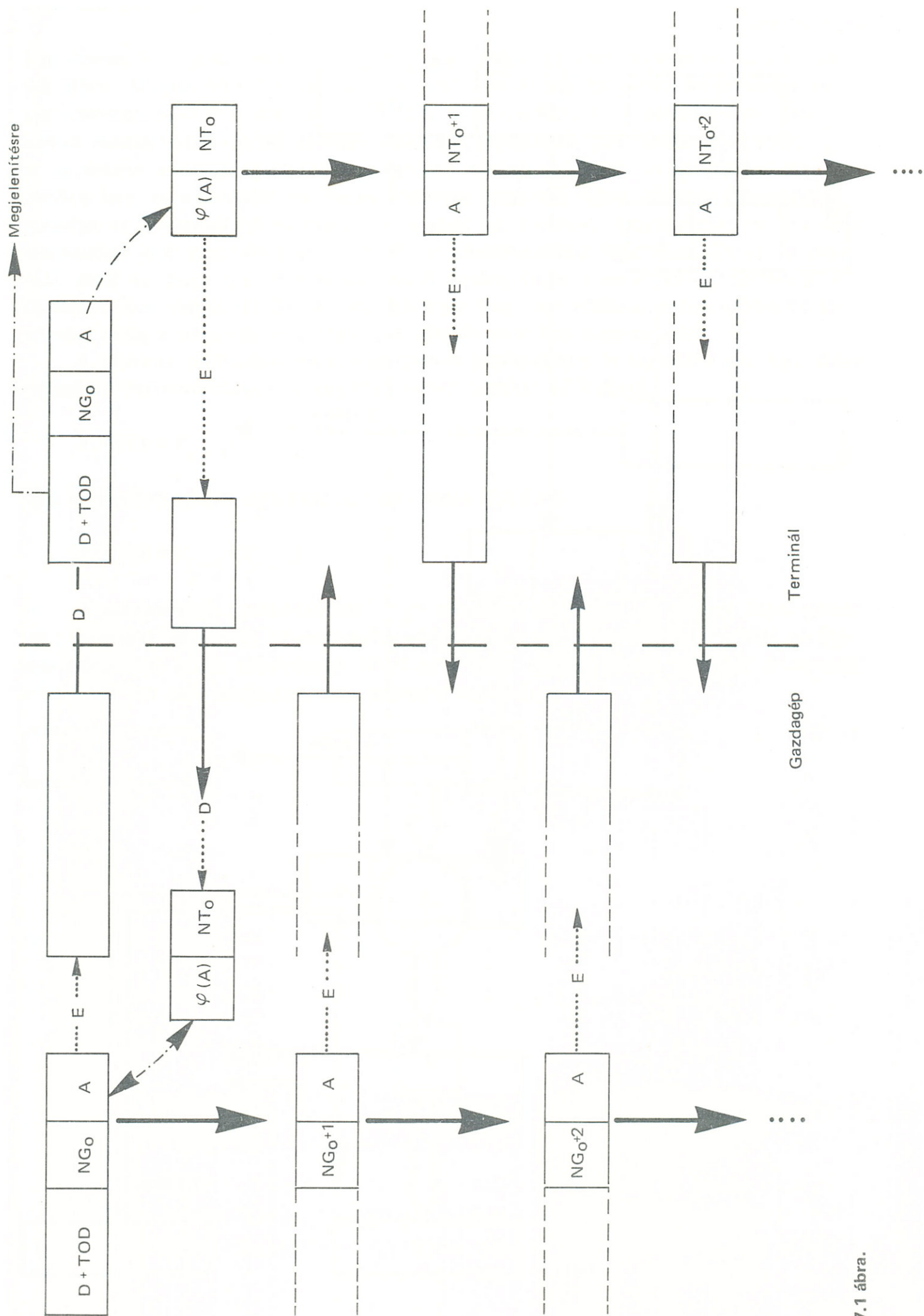
- egy üzenet az adott kapcsolathoz tartozik-e, és
- egy üzenet blokkjai eredeti sorrendjükben (felesleges vagy hiányzó blokkok nélkül) követik-e egymást.

7.1.1 Üzenetláncolás

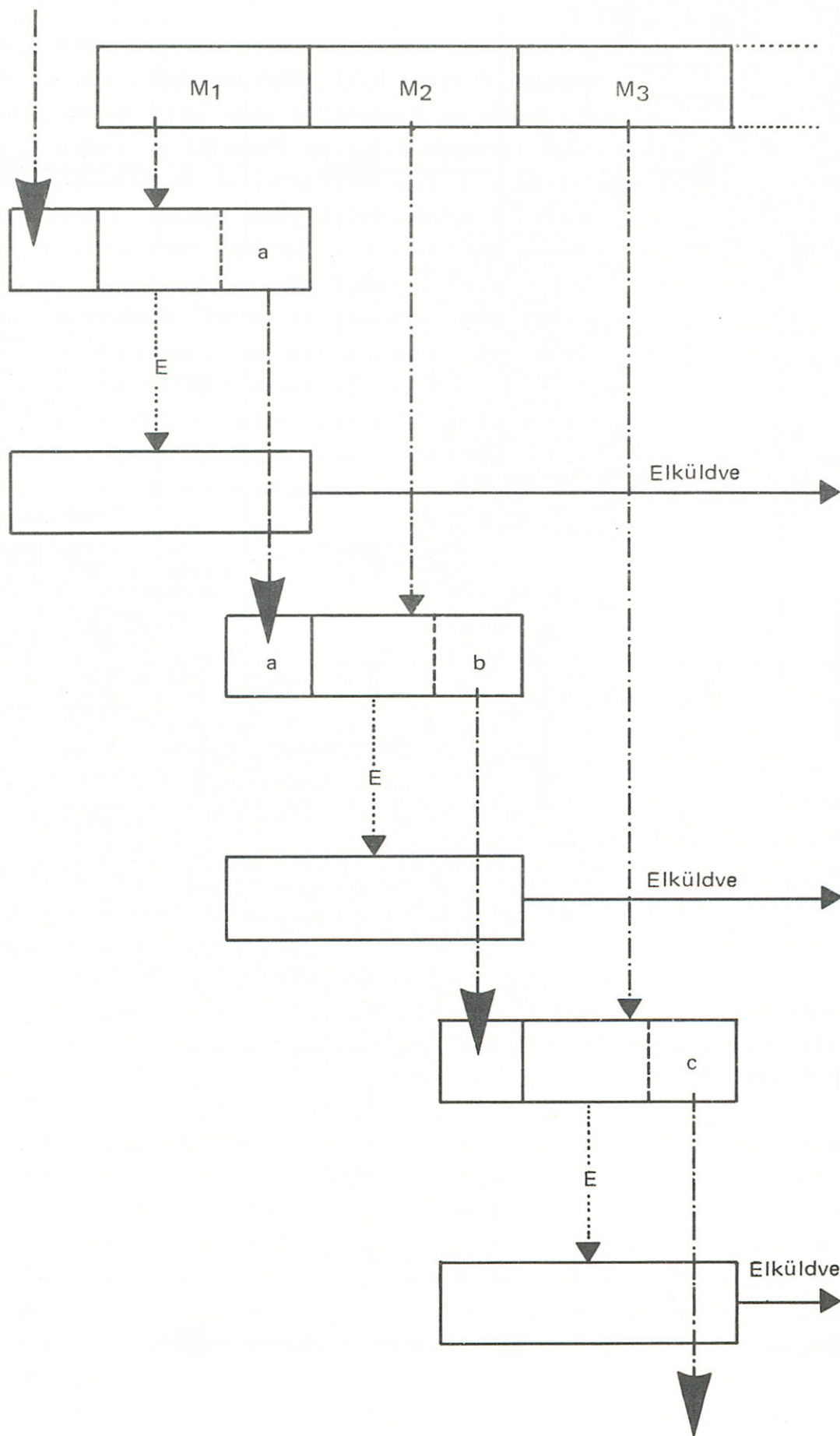
Egy kapcsolaton folytatott párbeszéd üzeneteinek sorrendiségét mindkét irányban **üzenetsorszám** alkalmazásával figyelhetjük. Az üzenetsorszámot az üzenetek kezeléséhez egyébként is szükséges vezérlőadatok között, az üzenetfejben helyezük el. Pontosabban az üzenetfejnek abban a részében, amelyet rejtjelezve továbbítunk. Az üzenetsorszámmal együtt az üzenetek egy **kapcsolatazonosítót** is hordoznak, amely az ugyanahhoz a kapcsolathoz tartozó üzeneteket rendeli össze. A 7.1 ábrán követhetjük végig a folytonos, „megbonthatatlan” kapcsolat kialakításának lépéseit, valamint az üzenet-visszajátszás elleni védekezés módját. Az események nyomon követését attól a ponttól kezdjük, amikor a partnerazonosítás (lásd 7.2 fejezet) már megtörtént, és a közös kulcs tudomásulvételével a védett kapcsolat kiépült.

Az egyszerűség kedvéért gazdagép-terminál kapcsolatot vizsgálunk, amelyben ezt a helyzetet a mesterkulcsok, illetve az alapkulcsrendszer felhasználásával érjük el. Feltételezzük továbbá, hogy a behatoló csak a terminál és a gazdagép közötti adatmozgatás útjába léphet be illegálisan.

A védett kapcsolat kialakítása után a gazdagép első üzenetében küldi a kapcsolat azonosítóját (A), a kezdeti üzenetsorszámot (NG_0) és velük együtt a hitelesítés céljait szolgáló pillanatnyi dátumot és rendszeridőt ($D + TOD$). A terminál ez utóbbit meg is jeleníti. A felhasználó $D + TOD$ alapján győződhet meg arról, hogy nem a behatoló játszik-e vissza egy korábbi gazdagép-üzenetet. A terminál első üzenetében a kapcsolatazonosító egy előre meghatározott függvényét, $\varphi(A)$ -t és a saját kezdeti üzenetsorszámát (NT_0) küldi vissza. Ez a nyugtázás a gazdagépet is meggyőzi arról, hogy az üzenet valóban a termináltól és nem a hálózatra beépült behatolótól származik. Ezt követően mindkét irány üzenetei az A kapcsolatazonosítót és a pillanatnyi üzenetsorszámot tartalmazzák.



7.1 ábra.



7.2 ábra.

7.1.2 Blokkláncolás

Egy üzenet blokkjainak sorrendi folytonosságát elvben az üzenetekéhez hasonlóan tudjuk elérni. Minden blokkban fenntartunk néhány bitet egy blokksorszám tárolására, és egy üzeneten belül a blokkokat folytatólagosan sorszámozzuk. A sorszámozásra fenntartott **rendszermező** mérete azonban csak igen korlátozott (pl.: 8 bit) lehet, mert az egyébként is eléggé rövid blokkok hasznos méretét csökkenti. A blokkszámozás így ciklikus lesz, és a behatoló egy teljes ciklusnak megfelelő számú blokkot észrevétlenül kivonhat az üzenetből (és esetleg be is iktathat az üzenetbe). Ugyanakkora méretű rendszermezővel jobb eredményt érhetünk el, ha **blokkláncolás**ra használjuk. Ennek lényege (7.2. ábra) az, hogy egy nyílt blokk utolsó néhány bitjét a soron következő blokk rendezőmezőjében megismétljük. Az így kialakuló lánc nem ciklikus, a rendezőmezők tartalmára pedig a behatoló a rejtett blokk alapján nem tud következtetni.

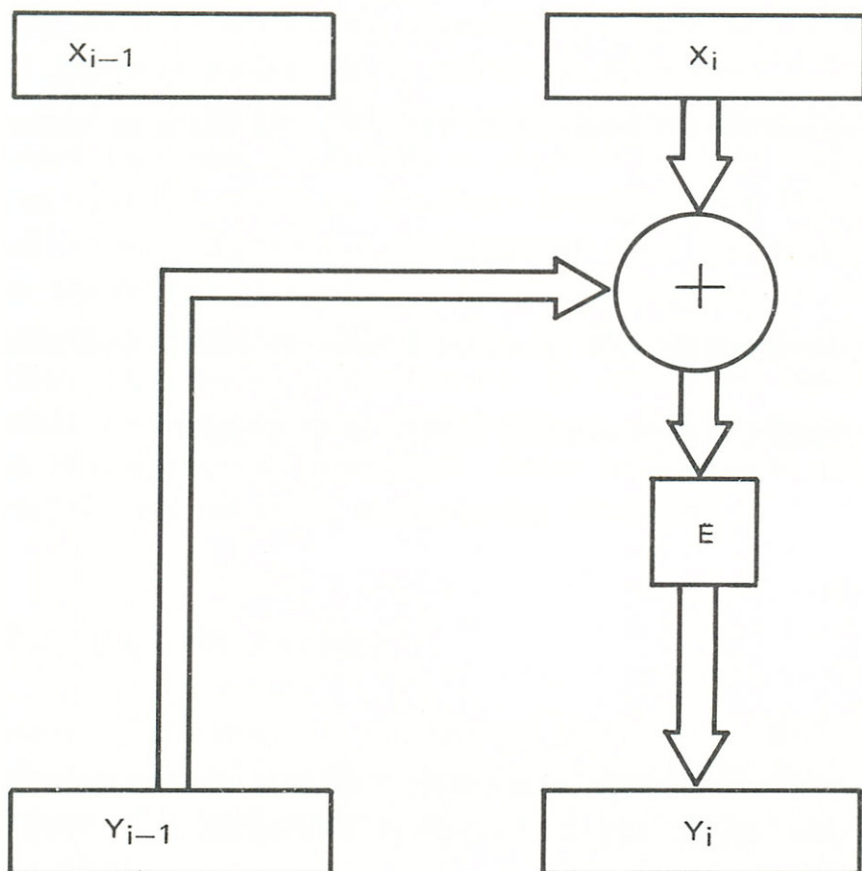
A blokkok sorfolytonosságát **visszacsatolt blokkrejtéssel** is biztosíthatjuk. Egy blokk rejtéséhez visszacsatolhatjuk a megelőző rejtett blokkot (7.3 ábra)

$$Y_i = E(X_i \oplus Y_{i-1})$$

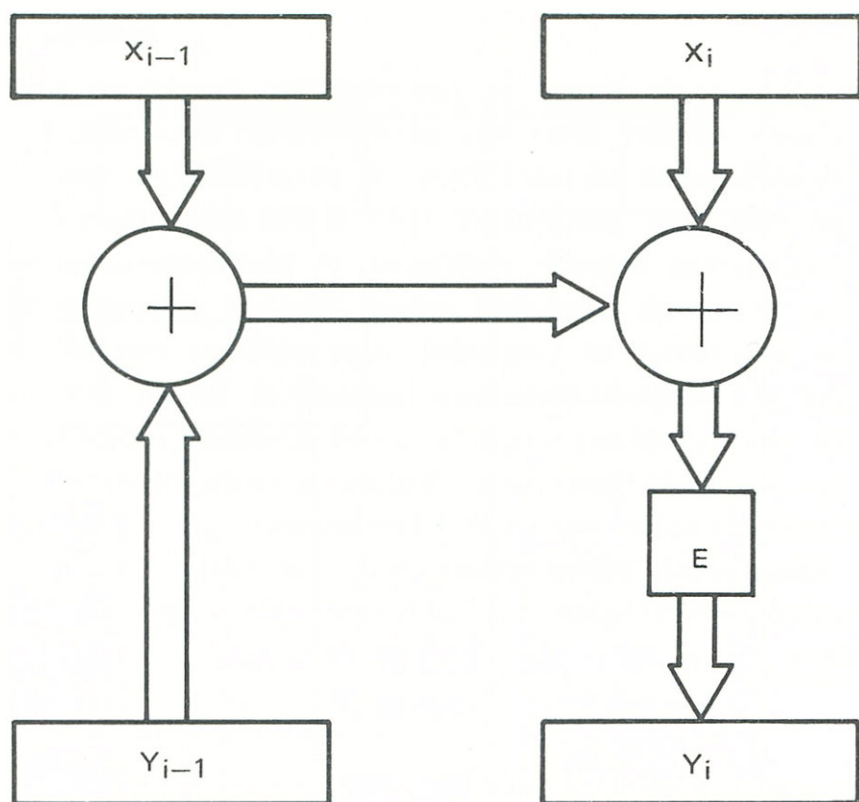
vagy a megelőző nyílt és rejtett blokkot együttesen (7.4 ábra)

$$Y_i = E(X_i \oplus Y_{i-1} \oplus X_{i-1}).$$

A visszacsatolás „beindításához” Y_0 -t megállapodásszerűen egy állandó bitmintának (pl.: zérusnak) választjuk. Visszacsatolás alkalmazásakor bármelyik blokk rejtését az összes előtte levő blokk befolyásolja. Mivel ez a hatás rejtések láncán keresztül érvényesül, a behatoló azt sem megfejteni, sem utánozni nem képes.



7.3 ábra.



7.4 ábra.

Az üzenethez utolsó blokként (X_{M+1}) egy előre megadott bitmintát (pl.: 101010...) tartalmazó blokkot csatolunk, majd ezt a teljes üzenetet a blokkvisszacsatolás valamelyik előbbi formáját választva rejtjük. A vétel helyén az egyes blokkok megfejtése

$$X'_i = D(Y'_i) \oplus Y'_{i-1}$$

vagy

$$X'_i = D(Y'_i) \oplus Y'_{i-1} \oplus X'_{i-1}$$

természetesen szintén függ az előzőleg vett rejtett blokkoktól ($Y'_0 = Y_0$). Az üzenetet akkor vettük sértetlenül, ha

$$X'_{M+1} = X_{M+1},$$

vagyis az utolsó megfejtett blokk tartalma megegyezik az eredeti üzenethez utolsó blokként csatolt bitmintával.

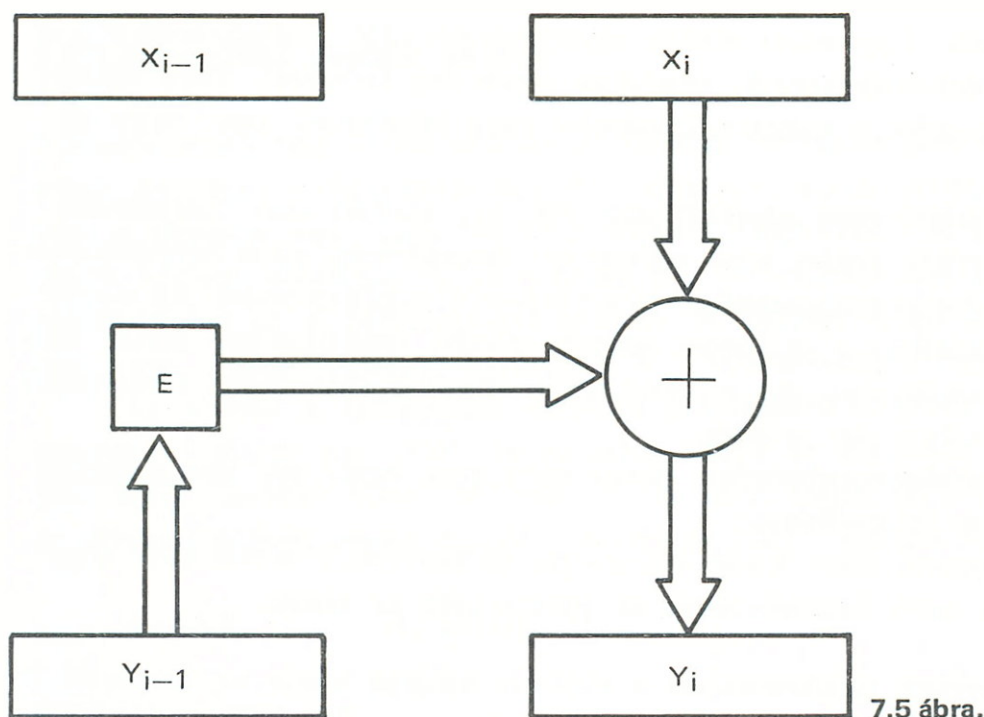
Blokk-visszacsatolással előállíthatunk kulcsfolyamos rejtést is (7.5 ábra)

$$Y_i = X_i \oplus E(Y_{i-1}),$$

amelynek megfejtése

$$X_i = Y_i \oplus E(Y_{i-1}).$$

Blokk-sorfolytonosság ellenőrzésére ugyanúgy használható, mint a visszacsatolt blokk-rejtés, mert a kulcsfolyam – és ezzel együtt egy blokk helyes megfejtése is – függ a továbbított blokkoktól.



7.1.3 Hibakorlátozás

A blokk-visszacsatolás alkalmazásának az az eredménye, hogy a rejtett üzenet bármelyik blokkjának meghibásodása az összes rákövetkező blokk megfejtését lehetetlenné teszi. Az üzenetek sértetlenségének ellenőrzését éppen a hibák hatásának ez a „terjedése” teszi lehetővé. Az utolsó, eredetileg egy megállapodás szerinti állandót tartalmazó blokk megfejtés utáni tartalma az üzenetben bárhol bekövetkezett meghibásodást kimutatja. Ezzel nemcsak a behatoló kártevési kísérlete leplezhető le, hanem a véletlenszerűen bekövetkezett meghibásodás is jelezhető. Tárolt adatok esetén azonban a hibajelzés önmagában még nem elegendő. Az adatvesztés elkerülésére hibajavítást is alkalmaznunk kell. A kérdés csak az, mi legyen a hibajavítás és rejtjelezés sorrendje.

Egy rejtjelező eljárás alapvető tulajdonsága, hogy egy nyílt-rejtett blokkpár bármelyikének legkisebb módosítása is párjának jelentős megváltozásával jár. Ugyanakkor a hibajavító kódolás módszerei általában kisszámú hiba hatásának megszüntetésére alkalmasak. Ebből következik, hogy a *rejtjelezésnek meg kell előznie a hibajavító kódolást*; vagyis **külső hibajavítást** alkalmazunk. Bármely hibakorlátozó kódolási eljárás számára vannak azonban felismerhetetlen hibaminták. Mivel ezeket az intelligens behatoló felhasználhatja a külső hibavédelem megkerülésére, az ismert **belső** (rejtjelezés előtt alkalmazott) **hibajelzés** használatát nem nélkülözhetjük. A két eljárás közötti „munkamegosztás”: a külső hibavédelem az adattároló, illetve adatmozgató közeg véletlen hibái, a belső hibajelzés pedig a behatoló tevékenysége ellen véd.

7.2 Digitális kézjegy

Iratokat, okmányokat mindennapi életünkben aláírással hitelesítünk. A jogilag rendezett eljárásokban, a személyes felelősség megállapításában, felhatalmazások és utasítások hitelességének ellenőrzésében mind-mind hasznosítják az aláírást, illetve annak néhány tulajdonságát.

- Az aláírás **nem hamisítható**. Egy adott aláírás egyértelműen utal „tulajdonosára”, és elválaszthatatlan attól a dokumentumtól, amelyhez eredetileg tartozott. (Egy aláírást nem lehet egy dokumentumról a másikra átmásolni vagy áthelyezni úgy, hogy az eredetinek látszék!)
- Az aláírás **felismerhető, tulajdonosa azonosítható**. Bár egy aláírást csak tulajdonosa képes előállítani, az érintettek széles köre egyszerűen felismerheti, vitás helyzetekben pedig lehetőség van a tulajdonos azonosságának hitelt érdemlő megállapítására. Az aláírás felismeréséhez és azonosításához a szükséges adatok mindig rendelkezésre állnak (aláírásminták, írásminták formájában). Az azonosításhoz azonban esetenként szakértőkre és speciális elemzési eljárásra van szükség.
- Egy aláírás **származása kétségbevonhatatlan**. Nincs mód arra, hogy egy dokumentum aláírója szerzőségét később letagadhassa.
- Az aláírás **előállítása egyszerű**. Nem kíván különleges és bonyolult eszközöket vagy eljárást. Az aláírás széles körű használatának az egyszerűség az alapja.

Az aláírást a számítógépes rendszerekben a **digitális kézjegy** váltja ki, amelytől lehetőség szerint ugyancsak a fenti tulajdonságokat követeljük meg. Mivel azonban a digitális kézjegy fizikailag nem kapcsolódik olyan szorosan a felhasználóhoz, mint az aláírás, néhány eltérés elkerülhetetlen. A legnagyobb különbséget az jelenti, hogy a digitális kézjegy egyediségét adó – egyébként titkosan kezelt – adato(ka)t tulajdonosa gondatlanságból vagy szándékosan nyilvánosságra hozhatja, illetve a behatoló eltulajdoníthatja. Ez lehetőséget nyújt arra, hogy tulajdonosa egy digitális kézjegy hitelességét kétségbe vonja. Külön intézkedések nélkül tehát a digitális kézjegy egy adott alkalmazásának hitelessége az egyediségét meghatározó titkos információk **teljes jövőbeli** védetségétől és sorsától függ.

A digitális kézjegyet meghatározó adatokat két csoportba sorolhatjuk: az előállítás és a hitelesítés adatai. Az előbbieket tulajdonosuk lehetőség szerint titkosan kezeli, a hitelesítő adatokat pedig a szükséges mértékig nyilvánosságra hozza. Egy digitális kézjegy ellenőrzése (tulajdonosának ismeretében) hitelesítő adatai segítségével történik, ezért **kétségbevonhatatlan azonosításuk** elsőrendűen fontos feladat. Egy behatoló ugyanis visszaélési lehetőségeket teremthet magának, ha egy digitális kézjegy hitelesítő adatait észrevétlenül sajátjának hitelesítő adataira képes kicserélni. A hitelesítő adatok felhasználására háromféle helyzetben lehet szükség.

- a) Egy dokumentum eredetét címzettje saját maga számára megnyugtatóan tisztázni akarja.
- b) Egy dokumentum eredetét címzettje valakinek hitelt érdemlően bizonyítani akarja.
- c) Egy dokumentum feltételezett szerzője a felelősséget – mások számára is bizonyított módon – el akarja magáról hárítani.

A digitális kézjegyeket két nagy csoportba sorolhatjuk. Ha a hitelesítő adatokból az előállítás adataira nem lehet következtetni, akkor a digitális kézjegy valódi, ellenkező esetben közvetített. A **valódi digitális kézjegy** előállítását és hitelesítését a felhasználók saját hatáskörükben végre tudják hajtani; a **közvetített digitális kézjegy** előállításához, illetve ellenőrzéséhez azonban a rendszer egy tökéletesen megbízható elemének közreműködése nélkülözhetetlen (mert a hitelesítő adatok nem hozhatók nyilvánosságra). A digitális kézjegy mindkét formájának magját rejtjelezési eljárás alkotja.

7.2.1 Közvetített digitális kézjegy

Konvencionális rejtjelezési eljárást alkalmazó rendszerekben az A felhasználó azonosításának általában mindig része egy K_A rejtjelezési kulcs, amelyet az A felhasználón kívül csak a rendszer egy (vagy néhány) teljesen megbízható és jól védett eleme ismer. A digitális kézjegy előállítás folyamatában K_A felhasználásával biztosítható, hogy a kézjegy egyértelműen az A felhasználóra utaljon. A kézjegy maga azonban csak közvetített lehet, mert hitelesíthetősége egyébként K_A nyilvánosságra hozását igényelné.

Az aláírást a felhasználó kérésére a **hitelesítő központ (HK)** készíti el. Az A felhasználó először egy $H(M)$ lenyomatot készít az aláírandó M üzenetről, majd egy aláírást kérő üzenetet küld HK -nak – 7.6 ábra. (Most és a fejezet hátra levő részében az áttekinthetőség végett $E_K(M)$ helyett az $\{M\}^K$ jelölést is használjuk.)

$$A \rightarrow HK: \quad A, \{H(M)\}^{K_A}. \quad (7.A.1)$$

Amit attól meg is kap:

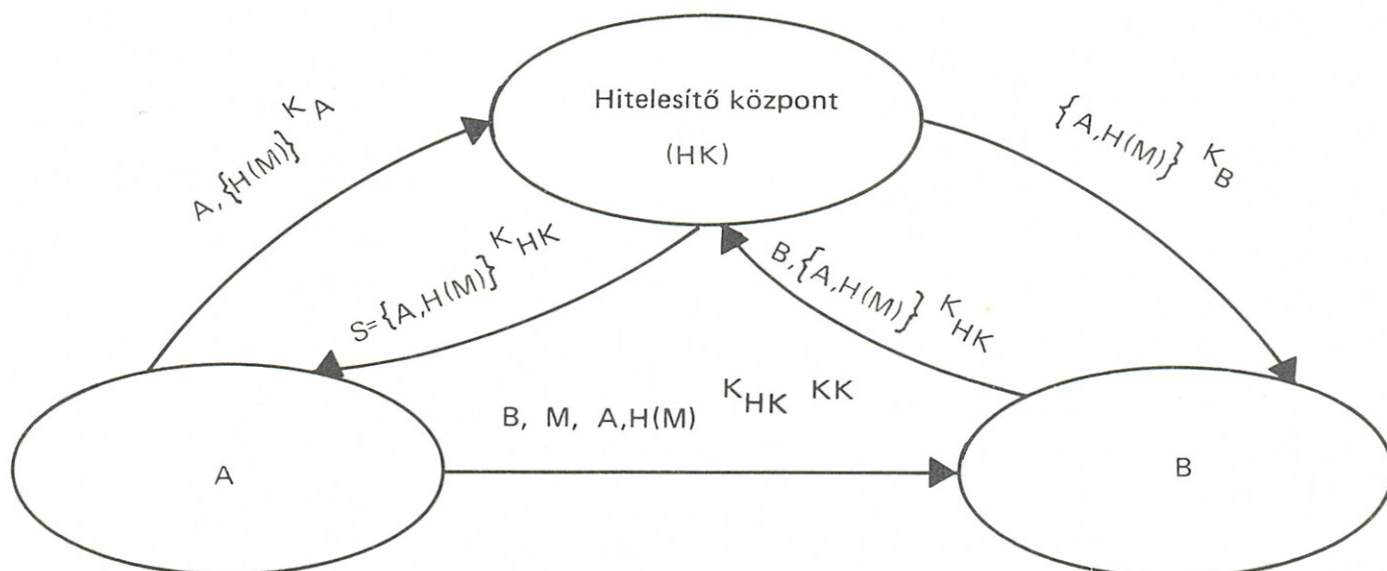
$$HK \rightarrow A: \quad \{A, H(M)\}^{K_{HK}} = S(A, M) \quad (7.A.2)$$

Ezt az üzenetet HK saját kulcsával rejti, ezért magát az aláírást – amely az $[A, H(M)]$ pár – még A sem tudja ellenőrizni. Ha azonban $S(A, M)$ -et a behatoló bármilyen módon megváltoztatja, A partnere – B – az M üzenetet nem fogja hitelesnek elfogadni. Az aláírás kézhezvétele után A az alábbi üzenetet küldi partnerének:

$$A \rightarrow B: \quad B, \{M, \{A, H(M)\}^{K_{HK}}\}^{K_K}. \quad (7.A.3)$$

Ha a megfejtés után B hitelesíteni akarja a vett M' üzenetet (amely M -től a továbbítás közbeni torzulás miatt eltérhet), először az aláírást elküldi HK -nak

$$B \rightarrow HK: \quad B, \{A, H(M)\}^{K_{HK}}. \quad (7.A.4)$$



7.6 ábra.

ahonnan már „érthető”, formában kapja vissza:

$$HK \rightarrow B: \quad \{A, H(M)\}^{K_B}. \quad (7.A.5)$$

Ezután elkészíti a vett M' üzenet $H(M')$ lenyomatát, és az összeveti a HK -tól kapott aláírásban szereplő $H(M)$ lenyomattal. Az üzenetet hitelesnek fogadja el, ha a lenyomatok megegyeznek:

$$H(M) = H(M') \Rightarrow M' = M$$

A későbbi vitás helyzetek rendezésére B M -mel együtt $\{A, H(M)\}^{K_{HK}}$ -t is megőrzi. Ez utóbbi segítségével (7.7 ábra) bármely harmadik, C személy is meggyőződhet M hitelességéről, a

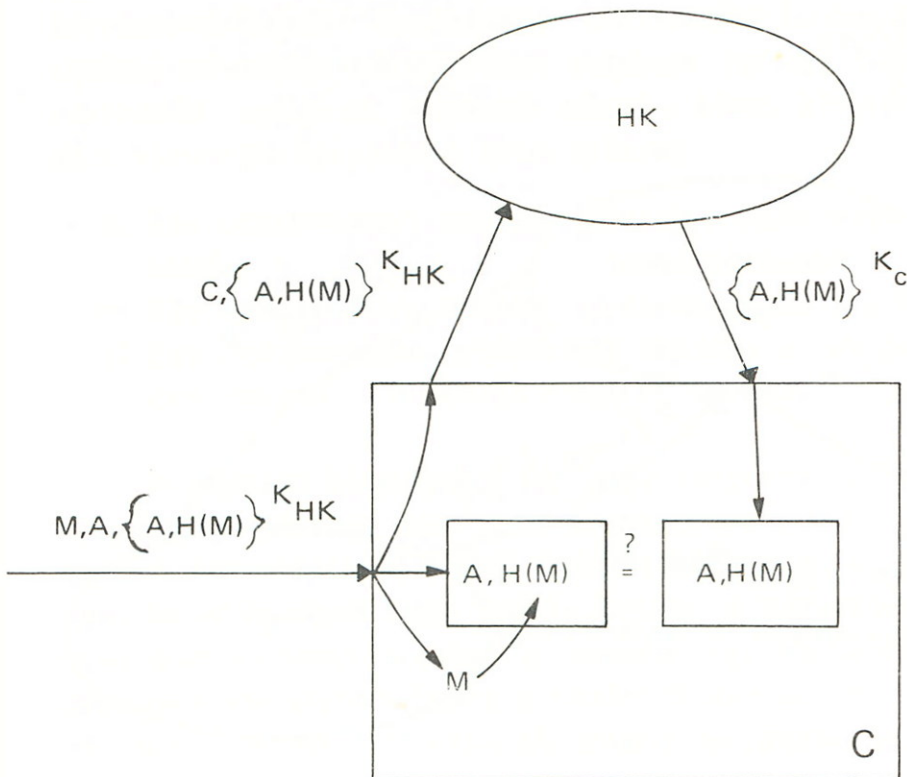
$$C \rightarrow HK: \quad C, \{A, H(M)\}^{K_{HK}} \quad (7.A.4')$$

$$HK \rightarrow C: \quad \{A, H(M)\}^{K_C} \quad (7.A.5')$$

üzenetváltás után (feltéve, hogy HK ismeri C -t).

Mindez hasonló ahhoz, ahogyan B meggyőződik az üzenet hitelességéről. Most azonban feltétlenül meg kell akadályoznunk bármelyik üzenet észrevétlen módosítását. Az A felhasználó ugyanis letagadhatná szerzőségét, ha az üzenetváltás útjába észrevétlenül beékelődhetne. Ezt természetesen kizárhatjuk az üzenethitelesítés már ismerttetett módszereinek alkalmazásával.

A teljes aláírási-hitelesítési folyamat áttekintése meggyőzhet bennünket arról, hogy az eljárás használhatósága az $M-H(M)$ kapcsolat „hamisítatlanságán” múlik. A kérdés ugyanis ez: $H(M)$ ismeretében tud-e B olyan M^* üzenetet előállítani, hogy $M^* \neq M$, de $H(M^*) = H(M)$?! Vegyük észre, hogy ha az aláírásban $H(M)$ helyett az M -et használtuk



7.7 ábra.

volna, ezzel a kérdéssel szembe se kerülünk, az aláírás mérete azonban összemérhető lenne magával az üzenettel! A lenyomat alkalmazásának éppen az a célja, hogy az aláírás az üzenet méretét általában ne növelje meg jelentős mértékben. Mivel a lenyomat mérete jóval kisebb az üzeneténél, ugyanahhoz a lenyomathoz számos (általában nem korlátos számú) üzenet tartozik. Az azonos lenyomatú üzenetek egymással való helyettesítését azonban meglehetősen nehezíti az a követelmény, hogy a helyettesítő üzenet is értelmes legyen. Ellenkező esetben a hamisításnak nincs sok értelme. Tovább nehezíthetjük a hamisító helyzetét, ha a hitelesített dokumentumokra kötelező formai megkötéseket teszünk (pl.: címzés, dátumozás, szövegezési fordulatok stb.), mert ezek csökkentik a beavatkozás szabadságfokát.

A lenyomatképző transzformáció nyilvánosan ismert. Ha azonban egyirányú függvénynek választjuk, a hamisítás **csak kísérletezéssel történhet**, mert $H(X)$ az azonos lenyomatú üzenetekre nézve semmilyen hasznosítható információt nem szolgáltat. Egyirányú függvényt sokféle módon előállíthatunk, mivel azonban az egész aláírási eljárás rejtjelezésen alapul, a következő megoldás kézenfekvő. Az M üzenetet a rejtjelezési rendszer kulcsméretével azonos méretű $M_1, M_2, \dots, M_L$ blokkokra bontjuk (DES esetén ez pl. 56 bit), majd egy nyilvános CC konstans (mérete a rejtjelezés blokkmérete) segítségével a lenyomatot a

$$H(M) = \{CC^M\} = \{ \dots \{CC\}^{M_1} \}^{M_2} \dots \}^{M_L}$$

formában képezzük.

Egy számítógépes hálózatban a közvetített digitális kézjegy előállítását egy hálózati NYILVÁNTARTÓ (HIVATAL) egyik feladatává tehetjük. (A NYILVÁNTARTÓ egyéb feladatai: felhasználóazonosítás, aktív felhasználók tevékenységének felügyelete, felhasználók rendszerbeli helyzetét meghatározó adatok tárolása stb.)

a) A felhasználó azonosságát a NYILVÁNTARTÓ helyi komponense még a kapcsolatba lépéskor megállapítja, és a kapcsolat tartama alatt megőrzi.

b) A felhasználó aláírandó üzenetét a NYILVÁNTARTÓ rendelkezésére bocsátja a címzett egyidejű megjelölésével.

c) A NYILVÁNTARTÓ (nem szükségképpen a helyi komponens) előállítja az aláírást, és az üzenethez csatolja. Az aláírt üzenetet a címzetthez továbbítja, és csak az aláírásra felhasznált rejtjelező kulcsot őrzi (és jegyzi) meg.

d) A címzett az üzenet vétele után az aláírt üzenet bemutatásával kérheti annak megállapítását, hogy az valóban a feltételezett szerzőtől származik-e. A későbbi viták hasonlóképpen rendezhetők.

A NYILVÁNTARTÓ használata feleslegessé teszi a rendszer és a felhasználó közötti specializált protokollok alkalmazását. További előnye, hogy megfelelő szervezéssel az aláírásban felhasznált kulcs védettsége is növelhető.

- Az aláírást rejtő kulcsot például a NYILVÁNTARTÓ-nak a tranzakcióban részt vevő komponensei szerint választhatjuk; minden komponenspárhoz egyedi kulcs tartozhat. Evvel egy kulcs használatának gyakoriságát csökkenthetjük.
- Az aláírást rejtő kulcsok védelmét a 6.3 fejezetben megismert részletekben tárolással fokozhatjuk. Ezzel egyidejűleg nő a rendszer rugalmassága és megbízhatósága is, az aláírást a NYILVÁNTARTÓ bármelyik komponense előállíthatja.

7.2.2 Valódi digitális kézjegy

Egyirányú permutáción (5. fejezet) nyugvó nyilvános kulcsú rejtjelezési eljárásokkal – mint a MIT-eljárás (5.1. fejezet) – elegáns digitális kézjegy-előállítási módszert valósíthatunk meg. Ha egy A felhasználó E_A elrejtést és D_A megfejtést használ, az M üzenetet

$$S(A, M) = D_A(M)$$

módon – az üzenet „előtorzításával” – írja alá. A címzett az üzenet szerzőjének ismertetében a nyilvánosan elérhető E_A rejtés

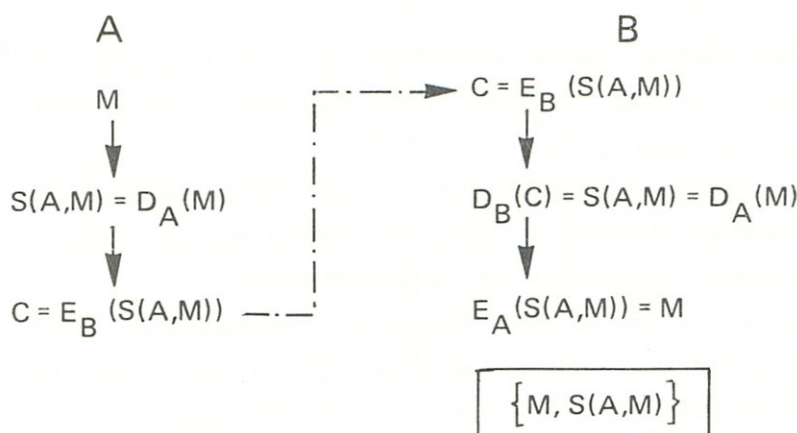
$$M = E_A(D_A(M)) = E_A(S(A, M))$$

alkalmazásával az üzenet birtokbavételével egyidejűleg annak eredetét is azonosítja. A teljes üzenettovábbítás – a 7.8 ábrán látható módon – az aláírást követően még a címzettnek szóló rejtést is magában foglalja.

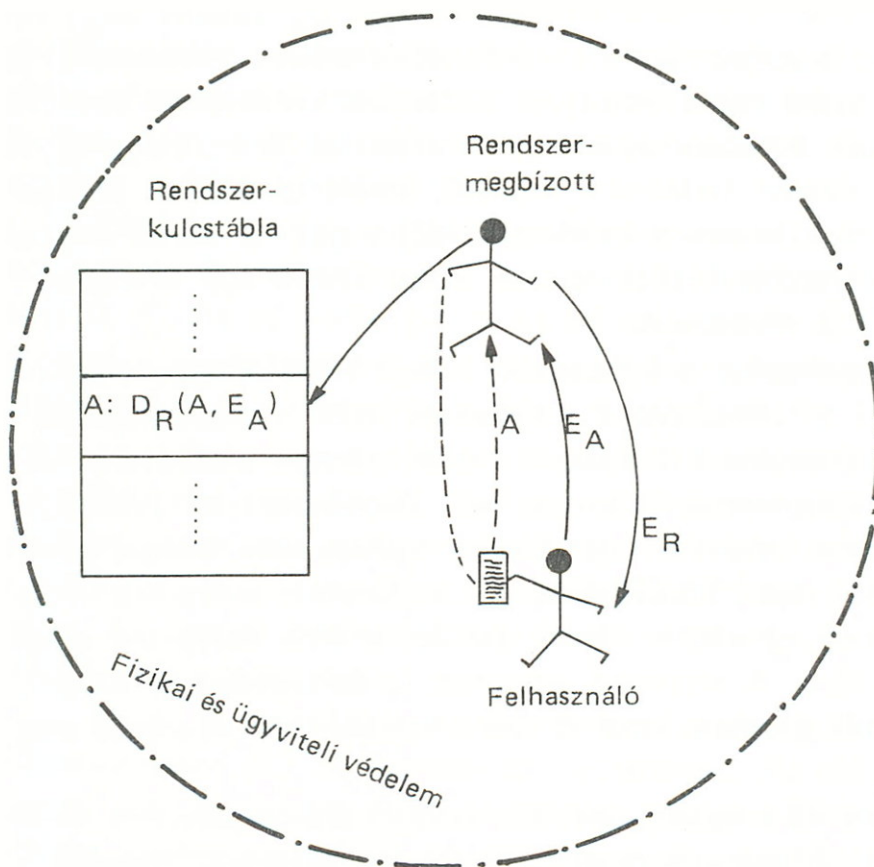
Ennek a digitális kézjegynek a hitelessége és a továbbítás védettsége egyaránt azon múlik, hogy a helyes (a feltételezett partnerhez tartozó) rejtéseket alkalmazzuk-e. Ha a behatolónak E_A -t sikerül észrevétlenül saját rejtésével helyettesítenie, B a behatoló üzeneteiről hiszi azt tévesen, hogy A -tól származnak; E_B kicserélése pedig M -et közvetlenül hozzáférhetővé teszi a behatoló számára. A rejtések (rejtőkulcsok) hitelességét azonban egy „rendszer aláírással” érhetjük el (7.9 ábra). Ennek érdekében minden felhasználó egyszer – a rendszer használói közé való felvételekor – a rendszeren kívül és biztonságos körülmények között kapcsolatba lép az üzemeltető megbízottjával. Ennek során

- a felhasználó átadja E_A rejtő transzformációját (illetve rejtőkulcsát);
- a megbízott a rendszeren kívül előállítja E_A rendszer aláírással ellátott $D_R(A, E_A)$ változatát, és elhelyezi a rendszerben;
- a felhasználó az ügynöktől átveszi a rendszer aláírás ellenőrzéséhez szükséges E_R rejtést (rejtőkulcsot);
- mindezt természetesen az ügyviteli védelem szabályainak megfelelően közösen dokumentálják. A behatoló ezután hiába lép be a felhasználó és a kulcstár közé, észrevétlen helyettesítést már nem tud végrehajtani.

A valódi digitális kézjegnek alapformájában a legnagyobb hiányossága az, hogy tulajdonosa érvénytelenítheti. Ehhez elég az aláíráshoz felhasznált adatok nyilvánosságra hozása vagy egyszerűen az eltulajdonításukra való hivatkozás. Sőt mi több az érvényte-



7.8 ábra.



7.9 ábra.

lenítés visszamenőleges hatályú, mert a vitás üzenet aláírásának időpontja nem ismert. Pedig az aláírás alkalmazásának pontosan az a célja, hogy az aláírt dokumentum tartalmaért az aláíró **utólag vissza nem vonhatóan** vállaljon felelősséget. A rendszer (kötelező) közreműködése tehát még a valódi digitális aláírás folyamatában sem kerülhető el.

Ismét a mindennapi gyakorlatból vett párhuzam alapján a rendszerben egy vagy több olyan KÖZJEGYZŐ-t működtetünk, amelyet feltétlenül megbízhatónak tekintünk. A KÖZJEGYZŐ-k elsődleges feladata, hogy a hozzájuk benyújtott aláírt üzeneteket visszavonhatatlan és meghamisíthatatlan időpont-megjelöléssel lássák el. Feladata ellátásához minden KÖZJEGYZŐ-nek egy saját (E_{KJ}, D_{KJ}) párja van. A felhasználó és a közjegyző közötti üzenetváltásban azonban az $E_U(M)$, illetve $D_U(M)$ helyett az $\{M\}^{PK_U}$ illetve az $\{M\}_{SK_U}$ jelölést használjuk. (PK_U a nyilvános rejtő-, SK_U a titkos fejtőkulcs).

Az A felhasználó az elküldendő üzenetét először aláírja, majd a közjegyzőhöz küldi:

$$A \rightarrow KJ: \quad \left\{ \left\{ M \right\}_{SK_A} \right\}^{PK_{KJ}}.$$

Ebből az üzenetből M -et KJ kicsomagolja, hozzácsatolja a pillanatnyi dátumot és időt ($D + TOD$), majd az eredményt a saját kulcsával aláírva visszaküldi A -nak:

$$KJ \rightarrow A: \quad \left\{ \left\{ \left\{ M \right\}_{SK_A}, D + TOD \right\}_{SK_{KJ}} \right\}^{PK_A}.$$

Ezt az üzenetet kapja meg a címzettje:

$$A \rightarrow B: \quad \left\{ \left\{ \left\{ M \right\}_{SK_A}, D + TOD \right\}_{SK_{KJ}} \right\}^{PK_B}.$$

Az A felhasználó ezt az aláírást tehát már nem érvénytelenítheti D_A (illetve SK_A) nyilvánosságra hozatalával, mert az aláírás időpontja hitelt érdemlően ismert. Hivatkozhat azonban arra, hogy digitális kézjegyet tudta nélkül eltulajdonították, és kiválasztott üzenetek hamisítására használták fel. Az egyik álláspont ezzel kapcsolatban az lehet, hogy egy felhasználó felelőssége minden olyan üzenet tartalmára kiterjed, amelynek aláírási dátuma megelőzi az aláírás érvénytelenítésének hivatalos bejelentési időpontját. A másik stratégia az aláírás körülményeinek még szigorúbb kézben tartása lehet. Ennek egy lehetséges módja egy rendszerbeli KÖZJEGYZŐ alkalmazása.

A KÖZJEGYZŐ elsőként megállapítja a felhasználó kilétét (lásd felhasználóazonosítás), majd ennek alapján beszerzi a felhasználó E_A nyilvános rejtő transzformációját (illetve nyilvános kulcsát). A rendelkezésre álló adatok alapján minden alkalommal ellenőrizheti, hogy az aláírást valóban tulajdonosa állította-e elő. Megkövetelhető például, hogy a felhasználó az aláírt üzenetbe meghatározott helyen foglalja bele a saját (rendszerbeli) azonosítóját. Ez a megoldás igen közeli rokona a közvetített aláírásnak. Fontos különbséget jelent azonban, hogy az aláírás ellenőrzéséhez, a vitás helyzetek rendezéséhez már nincs szükség közvetítőre. A hitelesítő adatokat (a felhasználó és KÖZJEGYZŐ nyilvános kulcsa) sem kell titkosan kezelni, ezért hosszú távú tárolásuk megbízhatóbb.

A másik megoldáshoz a terminálba épített speciális berendezéskomponensre és algoritmusra van szükség. A felhasználó titkos kulcsát (D_A -t) mágneskártya, integrált áramkör stb. tartalmazza, amelyet használatkor a terminál megfelelő fogadó részébe helyezünk. A felhasználó azonosítása után a terminál automatikusan elvégzi a kulcs ellenőrzését, és csak ezután engedélyezi (legalábbis aláírásra) a kulcs használatát. A kulcs eltávolításakor érvényessége megszűnik, és a kulcsellenőrzést újra el kell végezni. (A kulcs ellenőrzése pl.: a rendszerben tárolt E_A lekéréséből, egy RN véletlenszám generálásából és az $RN = E_A(D_A(RN))$ azonosság ellenőrzéséből állhat.)

7.2.3 Egyéb digitális aláírási módszerek

Lamport [25] és Rabin [27] az előzőektől eltérő, de egymás közt hasonló elvű digitális kézjegy-előállítás módszereket javasolt, amelyekben az aláírás az üzenetre egyedi rejtjelező kulcsok sorozatával történik.

Egy n bites üzenet aláírásához **Lamport algoritmus**a alapformájában $2n$ darab

$$k_1, K_1, k_2, K_2, \dots, k_n, K_n$$

rejtjelező kulcsot állít elő álvéletlenszám-generálással, és ezeket titkosan kezeli. Ezzel együtt a hitelesítés céljára az

$$u_1, U_1, u_2, U_2, \dots, u_n, U_n$$

és a megfelelő

$$\{u_1\}^{k_1}, \{U_1\}^{K_1}, \{u_2\}^{k_2}, \{U_2\}^{K_2}, \dots, \{u_n\}^{k_n}, \{U_n\}^{K_n}$$

blokk sorozatot is létrehozza. Ezeket előre megküldi a címzettnek, és egyidejűleg egy közösen elfogadott és megbízható NYILVÁNTARTÓ-ban is letétbe helyezi. A címzett felelőssége, hogy a felhasznált hitelesítési paraméterek valódiságáról meggyőződjék. (Pl.: egyeztet a NYILVÁNTARTÓ-val.)

Később, amikor a feladó az M üzenetet elküldi, aláírásként a fenti $2n$ darab kulcsból n -et az üzenethez csatol. Az aláíró kulcssorozatban az i -edik kulcs k_i , ha M i -edik bitje 0, K_i , ha az i -edik bit 1 értékű. A címzett az üzenet eredetiségét bitenként ellenőrizheti a rendelkezésére álló hitelesítő paraméterekkel: ha az üzenet i -edik bitjének értéke 0, akkor az aláírás i -edik kulcsa u_i -t $\{u_i\}^{k_i}$ -be, ha 1 akkor viszont U_i -t $\{U_i\}^{K_i}$ -be kell, hogy transzformálja. A címzett az aláírást hamisítani, illetve M bitjeit észrevehetetlenül megváltoztatni nem képes, mert a hitelesítő paraméterekből a kulcsok hiányzó felének egyetlen tagját sem tudja meghatározni. Vitás helyzetekben a címzett az aláírt üzenetet mutatja be, és hivatkozik a NYILVÁNTARTÓ-ban letétbe helyezett hitelesítő paraméterekre. Az üzenet hitelességét az aláírás helyessége, a szerző azonosságát pedig a NYILVÁNTARTÓ szolgáltatja.

Ebben a formájában az aláírás igen gazdaságtalan, mert hossza többszöröse (DES alkalmazásakor pl.: 56-szorosa) az üzenetének. Az aláírás azonban készülhet csupán az üzenet már megismert $H(M)$ lenyomata alapján. Ez az üzenethossztól független méretű aláírást eredményez, DES alkalmazása esetén $64 \times 56 = 3584$ bit hosszúságút. Tovább csökkenthetjük az aláírás méretét, ha $H(M) = \{CC\}^M$ -nek is csak az első c bitjét vesszük alapul. Mivel bármely blokkrejtés definíciószerűen jó álvéletlenszám-generátor is, közelítőleg 2^{-c} annak a valószínűsége, hogy $M' \neq M$ esetén $\{CC\}^{M'}$ és $\{CC\}^M$ első c bitje megegyeznek. Ezzel egyidejűleg kb. $2^c - 1$ kísérlet kell ahhoz, hogy olyan M' üzenetet találjunk, amelynek lenyomata első c bitjében megegyezik M lenyomatával. Az aláíráshoz felhasznált bitek számát tehát úgy választjuk meg, hogy $2^c - 1$ a kívánt értéket elérje.

Rabin eljárásának alkalmazásához egy üzenet szerzője szintén véletlenszerűen állít elő kulcsokat,

$$k_1, k_2, \dots, k_{2r},$$

amelyeket titokban tart. A felhasznált kulcsok számát biztonsági megfontolások határozzák meg, megválasztására később még visszatérünk. A kulcsok hitelesítésére ugyancsak nyílt és rejtett blokkok sorozata,

$$u_1, u_2, \dots, u_{2r},$$

illetve

$$\{u_1\}^{k_1}, \{u_2\}^{k_2}, \dots, \{u_{2r}\}^{k_{2r}}$$

szolgál. A szerző ezeket a címzettnek és egy közösen meghatározott NYILVÁNTARTÓ-nak is előre megküldi. A címzett felelőssége, hogy a kapott hitelesítő paraméterek valódiságáról meggyőződjék. Az M üzenet aláírását lenyomatának a korábban kiválasztott kulcsokkal képzett

$$\{\{CC\}^M\}^{k_1}, \{\{CC\}^M\}^{k_2}, \dots, \{\{CC\}^M\}^{k_{2r}}$$

sorozata alkotja, amelyet szerzője az üzenethez csatol.

A címzett az aláírás (és az üzenet) hitelesítéséhez a szerzőt felszólítja $k_{i1}, k_{i2}, \dots, k_{ir}$ kulcsainak megküldésére. (Az $1 \leq i_1 < i_2 < \dots < i_r \leq 2r$ indexeket véletlen sorsolással határozza meg.) Ezeket a kulcsokat a hitelesítő paraméterekkel ellenőrzi, majd az aláírás megfelelő elemeit az üzenetből velük előállítja. Az újonnan előállított és az aláírásban szereplő elemek egyezése hitelesíti az üzenetet. Ha bármelyik kulcs vagy aláírás-elem ellenőrzése sikertelen, a címzett az üzenetet nem fogadja el (illetve azt csak saját felelősségére teheti).

Vita esetén a szerző a döntőbírónak bemutatja a $k_1, k_2, \dots, k_{2r}$ kulcsokat, amelyeket az a NYILVÁNTARTÓ-ban őrzött hitelesítő paraméterekkel ellenőriz. Ha ennek során csak egyetlen kulcs is hamisnak bizonyul, a döntőbíró a szerzőt máris elmarasztalja. Ezután újra előállítja a vitás M üzenet aláírását és az eredményt összeveti az eredetivel. Ha csak r vagy kevesebb elem egyezik, a döntés: a címzett hamisítást kísérelt meg. Ha $r+1$ vagy több elem közös: a szerző egy általa küldött üzenetet kísérel meg kétségbe vonni. Nézzük meg, hogy ez a döntési szabály milyen körülmények között vezet felfedezhetetlen (bizonyíthatatlan) visszaélésekhez.

A címzett az M üzenet vétele után annak hitelesítéséhez r kulcs birtokába jut. Ezeket felhasználhatja egy saját készítésű M' üzenet aláírásának részleges hamisításához, mert az $\{\{CC\}^{M'}\}^{k_{i1}}, \{\{CC\}^{M'}\}^{k_{i2}}, \dots, \{\{CC\}^{M'}\}^{k_{ir}}$ aláírás-elemeket elő tudja állítani. Ahhoz azonban, hogy a szerzőt (a fenti döntési szabály alapján) elmarasztaltathassa, még legalább egy további $k_j \in \{k_{i1}, k_{i2}, \dots, k_{ir}\}$ kulcsot is meg kell ismernie a szerző eredeti $2r$ kulcsa közül. Ehhez csak az $(u_j, \{u_j\}^{k_j})$ és a $(\{\{CC\}^{M'}\}^{k_j}, \{\{CC\}^{M'}\}^{k_j})$ párok állnak rendelkezésre, amelyek — a rejtjelezéssel kapcsolatban tett általános feltételezésünkéből következően — nem javítják esélyeit a minden támpontot nélkülöző kísérlétezéshez képest. A hamisítás másik esélye az, hogy a címzett olyan, számára célszerű M' üzenetet képes szerkeszteni, amelynek lenyomata megegyezik M -ével. Mint már korábban láttuk, ez is elfogadhatóan kis valószínűségű esemény.

A szerző egy üzenetének későbbi letagadását úgy készítheti elő, ha az az aláírásnak csak r elemét állítja elő eredetileg kiválasztott és hitelesítő paraméterrel ellátott kulccsal, míg a további r aláírás-elemet pl.: véletlenszerűen választja. Kísérlete akkor jár sikerrel, ha a címzett az üzenet hitelesítésére pontosan azokat az aláírás-elemeket választja ki, amelyeket a szerző szabályosan állított elő. Ennek valószínűsége azonban — mivel a címzett az ellenőrzéshez megkért kulcsokat véletlenszerűen válogatja ki — csak $\binom{2r}{r}^{-1}$. Ez $r=20$ mellett $p=10^{-11}$ -t, és még $r=8$ mellett is legfeljebb $p=10^{-4}$ -t eredményez. Ha egy felhasználóval szemben naponta átlagosan egy hamisítást kísérelnek is meg, ezek közül átlagosan még ez utóbbi esetben is csak 30 évenként lesz egy sikeres!

Rabin módszerének előnyei és hátrányai egyaránt azzal kapcsolatosak, hogy minden üzenet aláírásához új kulcssorozat szükséges, és egy kulcs ismételten nem használható fel. A módszer előnye nyilvánvaló: egy aláírást meghatározó adatok eltulajdonítása, nyilvánosságra hozatala csak egyetlen üzenet hitelességét érinti. A nagy mennyiségű kulcshitelesítő paraméter megbízható tárolása azonban komoly nehézségek forrása lehet, mert egy-egy dokumentum hitelességét sokszor évtizedek múltán is tudni kell bizonyítani vagy cáfolni. A rendszer a hitelesítő paraméterek tárolását a felhasználókra háríthatja, ekkor azonban külön kell gondoskodni a hitelességükről. A célnak a konvencionális vagy nyilvános kulcsú rejtjelezési eljárásokon alapuló korábbi digitális kézjegyek bármelyike megfelel. A felhasználók a hitelesítő paramétereket aláírt formában kapják meg, és így vitás helyzetekben a hivatkozási alapot képező hitelesítő paraméterek valódisága is megállapítható. (Közvetített rendszer aláírás esetén azonban csak speciális rendszerelem(ek) segítségével.)

7.3 Partnerazonosítás

A bizalmas kapcsolat létrehozását kezdeményezője — nevezzük A -nak — egy olyan üzenettel indítja meg, amely

- csak a szándékolt partner — mondjuk B — számára érthető; és
- B kétséget kizáróan megállapíthatja róla, hogy A -tól származik.

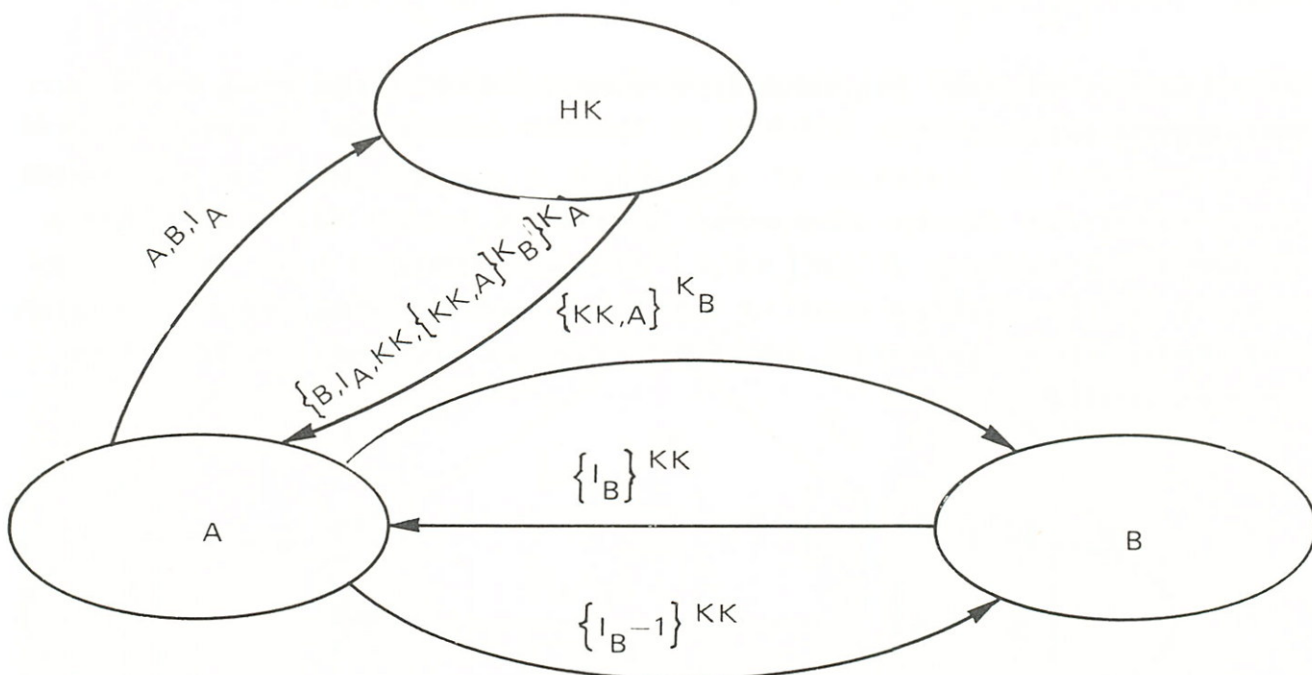
A kapcsolat felépítését B fejezi be azzal, hogy a kezdeményező üzenet tartalmát a saját maga azonosítására felhasználó üzenettel válaszol. Ezt követően mindkét fél az üzenethitelesítés (7.1 fejezet) már megismert módszereivel biztosíthatja a kapcsolat megamisíthatatlan folytonosságát. Világosan látnunk kell azonban, hogy ez — a kapcsolat felvétele során végrehajtott — **partnerazonosítás** nem a felhasználók azonosságának megállapítására szolgál. Feladata csupán az, hogy a kapcsolatot kereső felhasználók közé (akiknek az azonosítása már korábban megtörtént) a rendszeren belül ne léphessen be illetéktelen személy vagy rendszerelem. A kapcsolatfelvétel alatti partnerazonosítás nem helyettesíti a digitális aláírást sem, mert csak a kapcsolat tartamára és csak a kapcsolatban álló partnerek számára állapítja meg az üzenetek hitelességét.

7.3.1 Kapcsolatfelvétel konvencionális rejtjelezési eljárást alkalmazó rendszerekben

A kapcsolatfelvétel azzal kezdődik, hogy A nyílt üzenettel fordul a rendszer **hitelesítő központjához** (lásd 7.2 fejezet, közvetített digitális kézjegy), amelyben saját és kívánt partnere nevét (azonosítóját) és a sorsolt (többé nem használt) I_A ügyelet (tranzakció) azonosítót helyezi el (7.10 ábra).

$$A \rightarrow HK: \quad A, B, I_A. \quad (7.B.1)$$

A hitelesítő központ az üzenet vétele után előkeresi a két fél K_A , illetve K_B titkos azonosító kulcsát, és előállítja azt a KK kapcsolatkulcsot, amelyet A és B a kapcsolat-



7.10 ábra.

felvétel után használni fog. (Feltéve, hogy a kapcsolatfelvétel sikeres.) Mindezeket egy meglehetősen bonyolult üzenetbe foglalja bele:

$$HK \rightarrow A: \quad \left\{ B, I_A, KK, \left\{ KK, A \right\}^{K_B} \right\}^{K_A}. \quad (7.B.2)$$

Az üzenetet csak A képes megérteni, aki a megfejtés után a KK kapcsolatkulcs birtokbavételén kívül ellenőrzi, hogy a kezdeményező üzenetben szereplő partner- és ügynökazonosítót kapta-e vissza. A válaszban mindkét azonosítót meg kell ismételni. Ha a címzett neve — B — kimaradna, a behatoló a (7.B.1) üzenetben B -t saját (vagy bűntársa) nevére cserélhetné ki. Ennek következtében B helyébe a behatoló által választott partner léphetne. Ha viszont az ügyletazonosító maradna ki, a behatoló visszajátszhatna egy korábbi $A - B$ kapcsolatból lehallgatott (7.B.2) üzenetet, és ezzel egy korábbi kapcsolatkulcs újra felhasználását kényszeríthetné ki.

Miután A a KK kapcsolatkulcsot megjegyezte, a válasz K_B -vel rejtett részét továbbküldi:

$$A \rightarrow B: \quad \left\{ KK, A \right\}^{K_B}. \quad (7.B.3)$$

Ezt az üzenetet egyedül B tudja megfejteni, aki viszont a megfejtés után megismeri a közös kapcsolatkulcsot. Egyidejűleg hitelt érdemlően megismeri partnerét is. (Most látható, hogy a behatoló miért nem ér el eredményt, ha a (7.B.1) üzenetben A nevét a sajátjára vagy bűntársáéra cseréli le.) B azonban nem lehet biztos abban, hogy ez az üzenet nem egy korábbi (7.B.3) üzenet visszajátszása-e. Erről egy sorsolt (és többé nem használt) I_B azonosító elküldésével győződik meg,

$$B \rightarrow A: \quad \left\{ I_B \right\}^{KK}, \quad (7.B.4)$$

amelyre megállapodásszerű — pl.: $I_B - 1$ — választ vár.

$$A \rightarrow B: \quad \left\{ I_B - 1 \right\}^{KK}. \quad (7.B.5)$$

Miután ez a válasz is helyesen megérkezett, mindkét fél kellő bizonyosság birtokában lényegi párbeszédbe kezdhet.

A teljes protokoll öt üzenetből áll, ezek közül A megtakaríthatja az első kettőt, ha rendszeres partnereivel minden alkalommal ugyanazt a kulcsot használja. Ehhez a (7.B.2) üzenetből származtatott $B: (KK, \left\{ KK, A \right\}^{K_B})$ alakú tételeket tárol, és a kapcsolat felépítését a (7.B.3) üzenettel kezdi. A KK kulcs ismételt felhasználása miatt azonban, a visszajátszás elleni védekezésül, kétirányú nyugtázás szükséges. A (7.B.3) és (7.B.4) üzenetet például

$$A \rightarrow B: \quad \left\{ KK, A \right\}^{K_B}, \left\{ I'_A \right\}^{KK}, \quad (7.B.3')$$

$$B \rightarrow A: \quad \left\{ I'_A - 1, I_B \right\}^{KK} \quad (7.B.4')$$

helyettesítheti.

7.3.2 Kapcsolatfelvétel nyilvános kulcsú rejtjelezési eljárást alkalmazó rendszerekben

A kapcsolatfelvétel itt is nyílt üzenettel kezdődik (7.11 ábra):

$$A \rightarrow HK: \quad A, B \quad (7.C.1)$$

A válasz,

$$HK \rightarrow A: \quad \{B, PK_B\}_{SK_{HK}} \quad (7.C.2)$$

nemcsak B nyilvános rejtőkulcsát, de nevét is HK aláírásával hitelesítve tartalmazza. Ez utóbbi nélkül a beható a (7.C.1) üzenetben B nevét sajátjával vagy bűntársával helyettesíthetné, és ezzel a kapcsolatot maga, illetve bűntársa számára építhetné ki. B nyilvános kulcsának ismeretében A kapcsolatot kezdeményező üzenetet küld:

$$A \rightarrow B: \quad \{A, I_A\}^{PK_B}. \quad (7.C.3)$$

Ez az üzenet, amelyet csak B ért meg, közli B -vel, hogy egy A nevű felhasználó keres vele kapcsolatot, aki a kapcsolat azonosításában az I_A ügyletazonosítót fel akarja használni. B az üzenet megfejtése után megszerzi A nyilvános kulcsát:

$$B \rightarrow HK: \quad B, A, \quad (7.C.4)$$

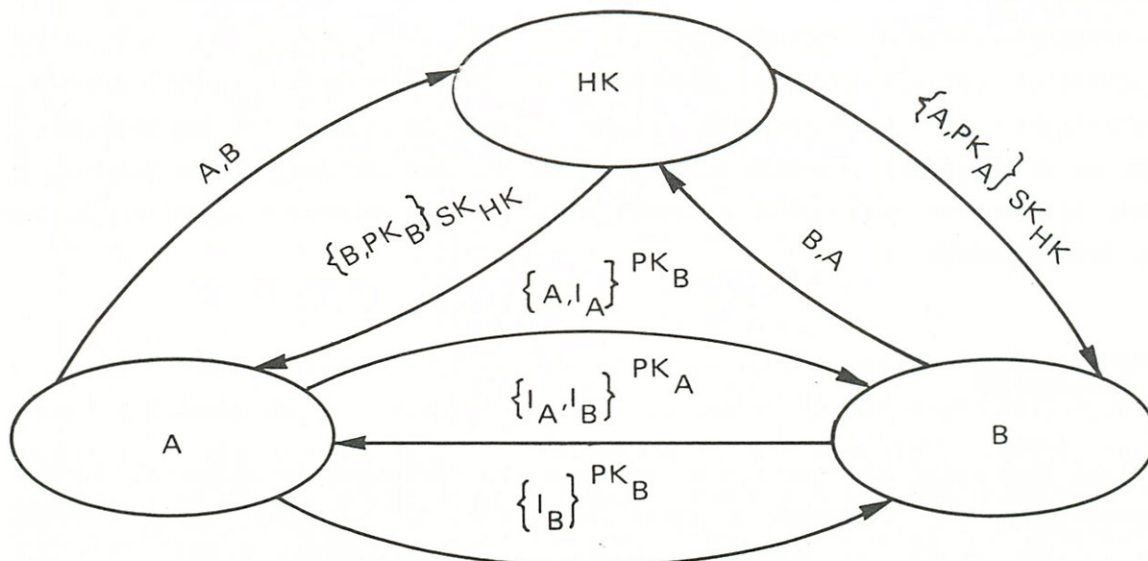
$$HK \rightarrow B: \quad \{A, PK_A\}_{SK_{HK}} \quad (7.C.5)$$

Ezután a visszajátszás elleni védelmet szolgáló kétirányú nyugtázás következik:

$$B \rightarrow A: \quad \{I_A, I_B\}^{PK_A} \quad (7.C.6)$$

$$A \rightarrow B: \quad \{I_B\}^{PK_B}. \quad (7.C.7)$$

Ebben a kapcsolatfelvételi protokollban tehát hét lépés szerepel a korábbi, konvencionális rejtjelezést alkalmazó rendszer öt lépésével szemben. Az üzenetek azonban most



7.11 ábra.

is három csoportba sorolhatók: kapcsolatkulcs-beszerezés, kapcsolatkezdemenyezés, nyugtázás. Nyilvános kulcsú rendszerben azonban a kapcsolat két irányában különböző kulcsokat kell használni, és ezeket a partnerek külön-külön szerzik be. Az üzenetek száma szintén háromra csökkenthető, ha minden előfizető helyben tárolja gyakori partnerei nyilvános kulcsát. [Ehhez elegendő a (7.C.2), illetve (7.C.5) üzenetekből származó, aláírt (név, kulcs) párok tárolása.]

A kapcsolatfelvételi üzenetek száma nemcsak helyi tárolással csökkenthető. Egy A felhasználó saját maga tárolhatja a rendszer által hitelesített nyilvános kulcsát: $\{A, PK_A\}_{SK_R}$. Ezt PK_R -rel együtt a rendszerhez csatlakozáskor szerzi be (lásd. 7.2 fejezet, valódi digitális kézjegy). A kezdeti üzenetváltás pedig ez lehet (7.12.a ábra):

$$A \rightarrow B: \quad \{A, PK_A\}_{SK_R} \{I_A\}_{SK_A}, \quad (7.D.1)$$

$$B \rightarrow A: \quad \{I_A, \{B, PK_B\}_{SK_R}, I_B\}^{PK_A}, \quad (7.D.2)$$

$$A \rightarrow B: \quad \{I_B\}^{PK_B}. \quad (7.D.3)$$

Ha A az ügyletazonosítót (I_A) illetéktelenek elől rejtve akarja tartani, az üzenetváltás néglépésesre bővül (7.12.b ábra):

$$A \rightarrow B: \quad \{A, PK_A\}_{SK_R}, \quad (7.E.1)$$

$$B \rightarrow A: \quad \{I_B, \{B, PK_B\}_{SK_R}\}^{PK_A}, \quad (7.E.2)$$

$$A \rightarrow B: \quad \{I_B, I_A\}^{PK_B}, \quad (7.E.3)$$

$$B \rightarrow A: \quad \{I_A\}^{PK_A}. \quad (7.E.4)$$

A gyakori partnerek nyilvános kulcsának helyi tárolásával azonban visszaállítható a háromlépéses sorozat: csak az előző üzenetváltásban kell a (7.D.1) üzenetet az

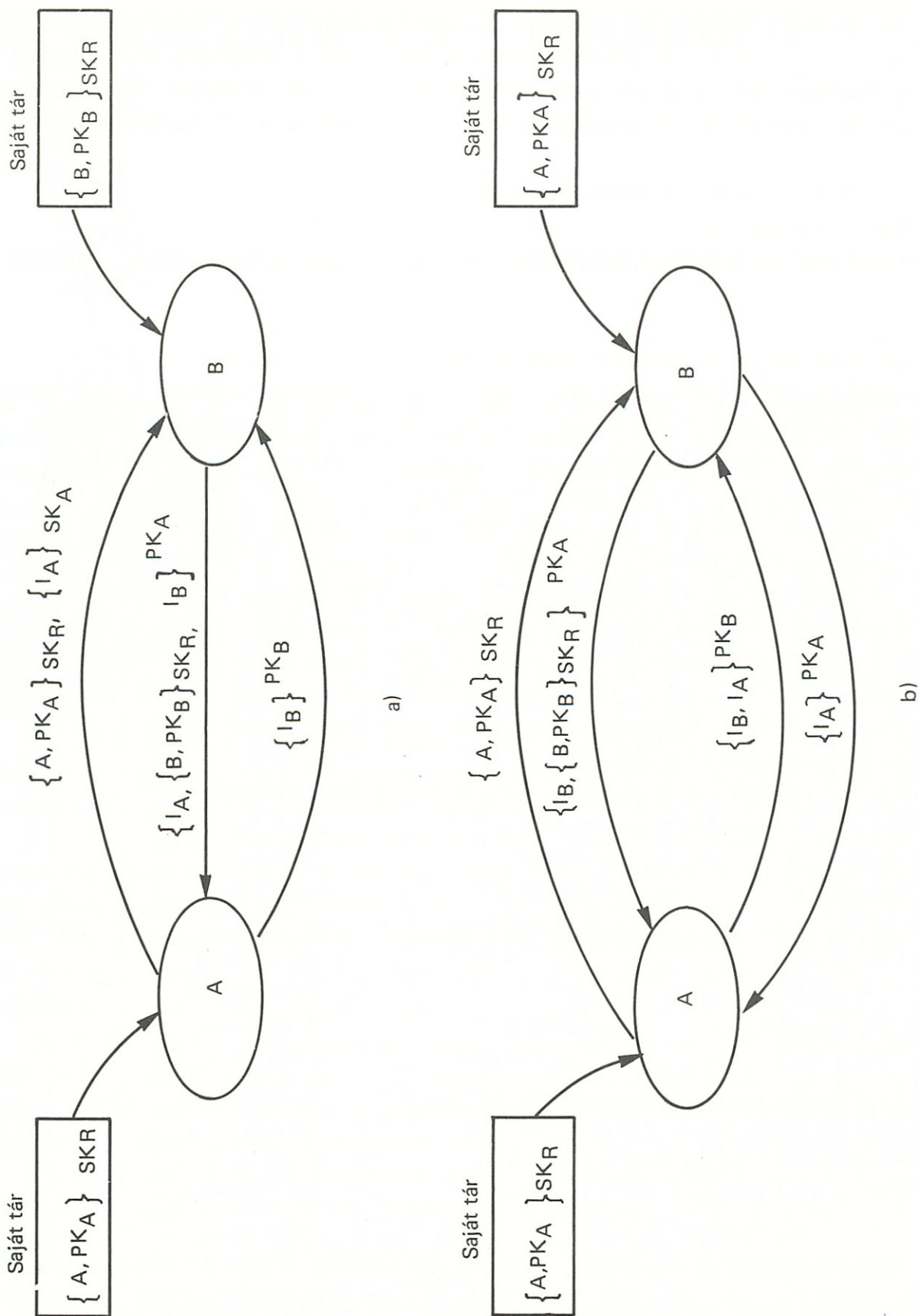
$$A \rightarrow B: \quad \{\{A, PK_A\}_{SK_R}, \{I_A\}_{SK_A}\}^{PK_B} \quad (7.D.1')$$

üzenettel helyettesíteni.

A hitelesítő központ megkerülésének azonban ára van. A nyilvános kulcsot tartalmazó hitelesített üzenetet kellő mértékben védeni és minden felhasználás előtt ellenőrizni kell, hogy az eltulajdonítást, az észrevétlen cserét megakadályozzuk. A legnagyobb problémát azonban az érvényüket veszített kulcsok visszavonásának megoldása jelenti. A felhasználók ugyanis hitelesített kulcsaikat kölcsön adhatják, és lehetnek egyetlen kulcsot használó csoportos felhasználók is.

7.4 Egyirányú kapcsolat

A közvetlen, kétoldalú kapcsolat mellett szinte minden számítógépes rendszernek fontos szolgáltatása az **üzenetkézbesítés**. Akárcsak a postai levélkézbesítő szolgálat, az üzenetkézbesítés is átveszi a feladó levelét, továbbítja és a címzett **postaládájába** helyezi, ahonnan



7.12 ábra.

a címzett később elviszi. A levél feladása és kézhezvétele között akkora idő telhet el, amely a feladó és a címzett között a közvetlen együttműködést kizárja. Ugyanakkor a feladó és a címzett kölcsönös azonosítása és a közvetített levelek visszajátzásának megakadályozása éppen olyan fontos feladat, mint a közvetlen kapcsolat esetén. Biztonságtechnikai megfontolások követelik, hogy maga a közvetítő ne játsszon szerepet a védel-

mi rendszer kialakításában. A védelem végleges formája természetesen függ attól, hogy konvencionális vagy nyílt kulcsú rejtjelezést alkalmazunk-e, de a megoldás elve azonos.

A levél egy **fejléct** kap, amelyet a levéltörzshöz az üzenethitelesítés tárgyalásakor megismert módszerek valamelyikével elválaszthatatlanul hozzáfűzünk. A fejlécnek hármas célja van:

- a) a levelet címzettje számára olvashatóvá teszi;
- b) azonosítja a feladót; és
- c) egy hamisíthatatlan **idő(pont)bélyegzést** tartalmaz a visszajátszás elleni védelem céljából.

Ha tartalma megkívánja, a levélben még digitális aláírás is lehet.

Az időpontbélyegzéshez nincs szüksége pontos szinkronizmust igénylő közös rendszeridőre, az **időpontbélyegző** bármelyik helyi komponense a saját átlagos pontosságú óráját használhatja. Az óra felbontóképességének azonban olyannak kell lennie, hogy ugyanazon felhasználótól származó két levél ne viselhesse ugyanazt a bélyegzett időt. Minden címzett (illetve levél fogadó ügynöke) feljegyzést vezet az érkező levelekről, amely (feladó, feladási időpont) alakú tételekből áll. Ugyanakkor minden egyes feladó-címzett párhoz tartozik egy-egy dT időkorlát; ez felső határa annak az időtartamnak, amely a levél feladása és megérkezése között eltelhet. A postaládát kezelő rendszerkomponens minden dT -nél idősebb vagy a feljegyzésben már szereplő (feladó, feladási idő)-vel rendelkező levél átvételét megtagadja. (Erről a levélközvetítő rendszer a levél általa ismert feladójának értesítést küldhet.) Kisméretű feljegyzést azzal biztosítunk, hogy a dT -nél idősebb tételeket rendszeresen töröljük belőle.

Konvencionális rejtjelezés használatakor a levél fejlécéként megfelel a

$$\{KK, A\}^{K_B}$$

(7.B.3) üzenet, a megelőző kapcsolatfelvételi protokollból, a teljes levél pedig (TS = küldési időpont)

$$\{KK, A\}^{K_B} \cdot \{TS, M\}^{KK}$$

Nyilvános kulcsú rejtjelezés használatakor a fejléc és üzenet

$$\left\{A, I \left\{B\right\}_{SK_A}\right\}^{PK_B} \left\{I, TS, M\right\}^{PK_B}.$$

Ebben $\{B\}_{SK_A}$ teszi lehetővé B számára, hogy A azonosítsa, a sorsolt I ügyletazonosító pedig a fejléc és levél összekapcsolására szolgál. (Az előbbi esetben ezt a láncolást közvetlenül KK biztosította.)

8. Hozzáférés – védelem

A rejtjelezésben az adatvédelem hatékony és az adathálózat védelmében nélkülözhetetlen eszközt ismertük meg. Természetesen annak sincs semmilyen elvi akadálya, hogy a rejtjelezést a számítógépen (pontosabban a számítógépes konfiguráción) belül is felhasználjuk adatvédelemre. Mint már korábban említettük is, a rejtjelezés természetes védelmi eszköze a mágnesszalagon tárolt információnak, de felhasználható a mágneslemez tárolók védelmében is. A megkívánt rejtjelezési sebességet mindig több rejtjelező berendezés egyidejű alkalmazásával (pontosabban több egymást követő blokk párhuzamos rejtjelezésével) érhetjük el. Van azonban néhány tény, amely arra figyelmeztet, hogy számítógépes rendszeren belül a rejtjelzés nem lehet a védelem egyetlen és kizárólagos eszköze.

1. A rejtjelezés alkalmas ugyan a felhasználók egymástól való elszigetelésére, nem teszi azonban lehetővé (legalábbis egyszerű eszközökkel és hajlékonyan nem) adatok közös és szabályozott birtoklását, felhasználását.

2. A rejtjelezéssel megakadályozhatjuk a védett adatok megismerését, de nem szabályozhatjuk felhasználásuk módját. Nem írható elő például az, hogy egy program kódját képező adat se megtekinthető, se módosítható ne legyen, csupán futtatható.

3. A rejtjelezett adatok védettsége a kulcs védettségének is függvénye. Mint már a 6. fejezetben is utaltunk rá, a kulcsvédelem végső soron csak a rejtjelező rendszeren kívüli módszerekkel valósítható meg.

4. Az egyébként rejtett formában tárolt és mozgatott adatok feldolgozása az esetek túlnyomó részében csak nyílt formában lehetséges. Multiprogramozott és időosztásos rendszerekben emiatt az egyidejűleg nyílt formában fellelhető adatok (programok) kölcsönös védelmére mindenképpen szükség van.

Ugyanakkor egy számítógépen belül valójában nincs is szükség az információ közvetlen rejtésére, a számítógép minden tekintetben jól védett közeg.

a) A benne tárolt információhoz csak meghatározott „csatornákon” át lehet hozzájutni. A számítógépterembe kevés (többnyire egyetlen) bejárat vezet, a termet magát általában a legmodernebb biztosító berendezések védik az egyéb behatolási utak igénybevételeivel szemben, a rendszer működésébe pedig csak néhány perifériája segítségével avatkozhatunk be.

b) A számítógép-konfigurációt érintő tevékenységek jól szabályozott rendszerbe szervezhetők. A gépteremben való tartózkodás nyilvántartással, a rendszer adathordozóinak (lemezcsomagok, mágnesszalagok) sorsa adminisztratív eszközökkel nyomon követhető és szabályozható.

A számítógép tehát **fizikailag** és **ügyvitelileg** is jól **védhető** közeg. A rendszerhez férés meghatározott érintkezési pontokra, a kapcsolat pedig meghatározott formákra korlátozható. Mindezek együttesen egy olyan belső, **algoritmikus védelem** lehetőségét teremti meg, amely a rendszerbe épülve és azzal együttműködve az érintkezési pontokon keresztül érkező kezdeményezések végrehajtását (általában kívülről meghatározott) keretek közé szorítja.

A számítógépen belüli algoritmikus védelem elsősorban nem az információk közvetlen rejtése hordozta védelmi lehetőségre támaszkodik, bár céljai érdekében ezt is felhasználhatja. Az információvédelmet hozzáférhetőségének és felhasználásának szabályozásával éri el.

8.1 Az adathozzáférés szabályozásának rendszermodellje

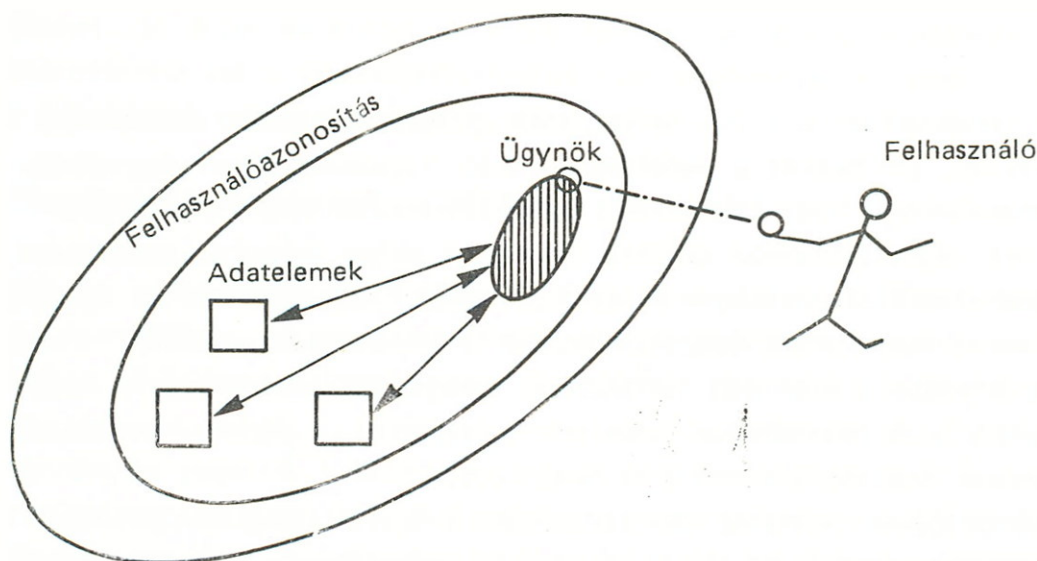
Az algoritmikus adathozzáférés-szabályozás teljes rendszerének vázát a 8.1 ábrán mutatjuk be. A rendszerbe az egyetlen út a **felhasználóazonosítás**on keresztül vezet. Ennek a rendszerelemnek (amelynek ismertetésére külön fejezetben térünk ki) a feladata, hogy a rendszert igénybe venni kívánó felhasználó személyazonosságát — legalábbis a rendszer védelméhez szükséges pontossággal — megállapítsa. Az azonosítás után a felhasználót a rendszerben a feladatait megoldó, igényeit a rendszerhez továbbító számítógépes folyamatok képviselik. Ezeket az adatvédelmi környezetben **ügynökök**nek nevezzük. Az ügynökök tevékenységüket **korlátozó közeg**ben végzik, amely viselkedésüket, lehetőségeiket kötelező erővel szabályozza. Többek között kikerülhetetlenné teszi a hozzáférést felügyelő rendszert, és érvényt szerez döntéseinek. A védett adattár felé irányuló minden igény jogosságáról és engedélyezhetőségéről ugyanis ez utóbbi rendszerrész dönt.

Azt, hogy a felhasználók a rendszert csak ügynökökön keresztül használhassák, azaz az algoritmikus védelmi rendszert ne kerülhessék meg, a **fizikai és az ügyviteli védelem** biztosítja. Ezek formáival és problémáival a továbbiakban nem foglalkozunk, két dolgot azonban, amelyek megoldása részben hatáskörükbe tartozik, tudatosítanunk kell.

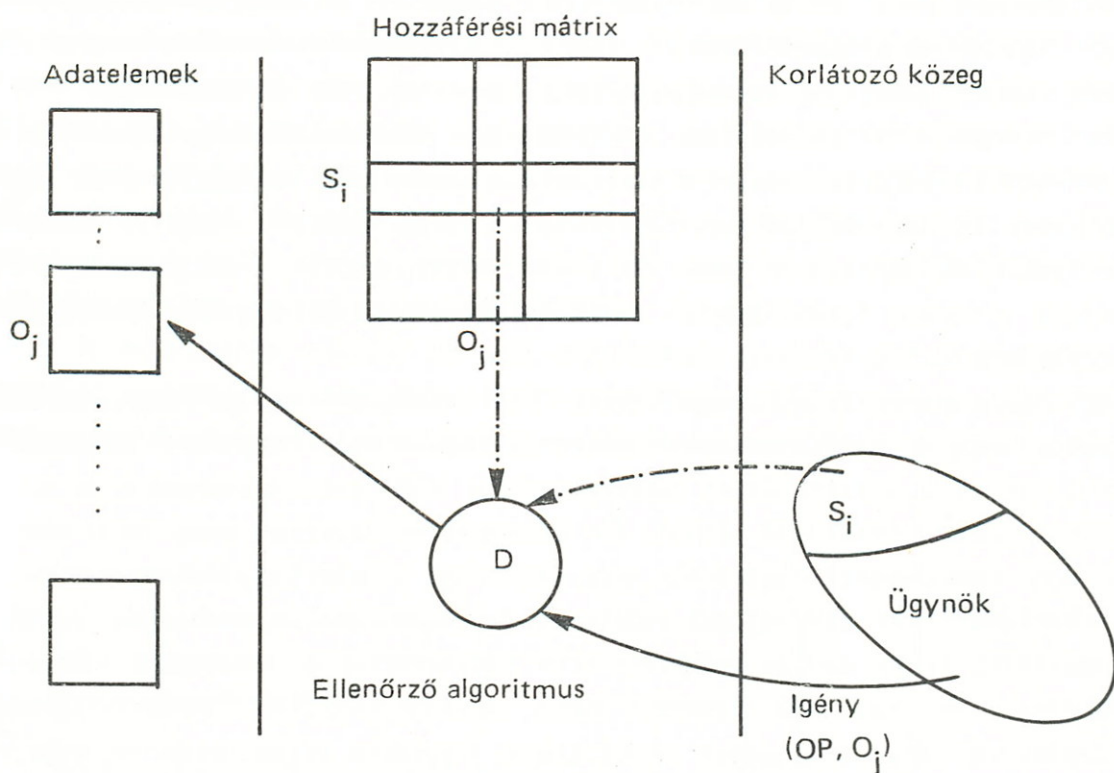
1. **Az algoritmikus védelem előfeltétele a megbízható felhasználóazonosítás.** Ha a felhasználó megszemélyesítése — azonosítására szolgáló ismereteinek (pl.: jelszó) és tárgyi kellékeinek (pl.: személyi igazolvány) eltulajdonításával — teljes mértékben a számítógépes rendszeren kívül történik, az algoritmikus védelem a behatolást nem képes megakadályozni. Ennek megakadályozása a fizikai és az ügyviteli védelem egyik legfontosabb feladata.

2. A teljes védelmi rendszer működésbe állításának és üzemeltetésének vannak olyan mozzanatai, amelyeket algoritmikus védelemmel nem szabályozhatunk. Magának az algoritmikus védelmet meghatározó rendszernek is vannak szükségszerűen olyan részei, amelyek működésének ellenőrzése kívül esik az algoritmikus védelem hatáskörén. (A további tárgyalás során ilyenekre példákat is találunk majd.) **A fizikai és elsősorban az ügyviteli védelem feladata tehát az is, hogy a rendszer szempontjából feltétlen bizalmat igénylő mozzanatok és rendszerelemek megbízhatóságát szavatolja.** Enélkül az algoritmikus védelem is teljesen hatástalanná válik.

Vegyük ezek után — a felhasználóazonosítás kivételével — kissé közelebből szemügyre az algoritmikus védelem összetevőit (8.2 ábra).



8.1 ábra.



8.2 ábra.

Adatelemek

A rendszerben tárolt információkból, adatokból a védelem céljainak és igényeinek megfelelő egységeket hozunk létre. Az egy-egy egységet, adatalemből alkotó információk, adatok a rendszerben azonos védelmet élveznek, a védelem szempontjából megkülönböztethetetlenek. Az adatalemből méretének megválasztásakor ellentétes követelményeket kell kielégíteni. Árnyalt védelmi stratégia nyilvánvalóan finom szemcsészettségű adatbázist, tehát kis adatalemből méretet kíván. Az adatalemből nyilvántartása és a védelmüket meghatározó információk tárolása azonban nagy adatalemből méret mellett gazdaságosabban valósítható meg (kevesebb „holt terhet” jelent). Tárolásuk ideje alatt az adatalemből oszthatatlan egységként viselkednek, feldolgozásuk közben azonban összetevőikhez is hozzáférhetünk. A védelem szempontjából ez is ellentétes követelmények együttes érvényesítésének igényével lép fel. Mindezekon kívül a védeleminformációt a „hozzáférési út mentén” kell elhelyezni, ezért a nyilvántartó és védelmi rendszer egyes pontjaikon szorosan összekapcsolódnak.

Az algoritmikus védelmi rendszerben a külső felhasználó igényeit az általa működésbe állított folyamatok közvetítik, amelyeket a védelemhez való kapcsolatukban **ügynökök**-nek nevezünk. (Ezzel hangsúlyozva, hogy feladatukat megbízás alapján látják el.) Egyetlen felhasználó feladatainak végrehajtásához számos ügynököt vehet igénybe, esetenként még párhuzamosan tevékenykedőket is. Alapvető szerepet játszik azonban az az ügynök, amelyet egy felhasználóhoz a rendszerbe lépésekor rendel a felhasználóazonosítás. Ennek az ügynöknek elsődleges feladata a képviselt felhasználó igényeinek közvetítése a rendszerhez. Mivel ez az ügynök a felhasználót mindenben helyettesíti, a védelmi rendszer további tárgyalása során csak ügynököket veszünk tekintetbe.

Az ügynökök a számítógépes rendszer „teremtmenyei”, tevékenységük ezért szigorú korlátok közé szorítható: minden adatelem-hozzáférési igényüket a védelmi rendszeren keresztül kell benyújtaniuk, és a döntéseit velük szemben a rendszer kötelező erővel érvényesíti. Az ügynökök (legalábbis a jól megtervezett) védelmi rendszert nem kerülhetik meg (még akkor sem, ha összejátszanak). Egy ügynök tevékenységének korlátait egyrészt annak a felhasználónak a lehetőségei határozzák meg, akinek a megbízásából tevékenykedik, másrészt maga a tevékenység, amelynek elvégzésére az ügynöknek „képesítése” van. Ez utóbbit az ügynök program része testesíti meg. A védelmi rendszer szempontjából az ügynököt tehát ez a két információ — a megbízó személye és a végrehajtott program funkciója — határozza meg, az ügynök lehetőségeit ezekből származtathatjuk.

Egy hétköznapi példa mindezt jól megvilágítja. Telefonkészülékek javítására bárhol a világon csak a posta (vagy a telefontársaság) alkalmazásában álló, megfelelő képesítésű műszerész jogosult, mivel a szakszerűtlen beavatkozás a központ működésére és a kábelhálózatra is káros következménnyel járhat. Ez önmagában azonban még nem elegendő ahhoz, hogy egy szakképzett műszerész egy elromlott telefonkészüléket megjavítson. Ehhez a telefonkészüléket üzemeltető előfizető hozzájárulása is szükséges. (Aki például ellenőrizni szeretné, hogy javítás ürügyén nem helyeznek-e el lehallgató készüléket telefonkészülékében.) A műszerész ilyenkor tehát mint a készülék üzemeltetőjének a telefonhálózat tulajdonosa által is elismert szakértelmű ügynöke végzi tevékenységét. Lényeges azonban azt is észrevennünk, hogy a telefont üzemeltető előfizető példánkban olyan feladatot képes ügynökén keresztül végrehajtani, amelyre egyébként önállóan nem jogosult.

Ellenőrző algoritmus

Minden, a rendszer védelmét potenciálisan érintő művelet csak ennek az ellenőrző algoritmusnak a közbeiktatásával fejtheti ki hatását. A célba vett adatelemet és a hozzáférési igényt (a kezdeményezett művelet megjelölésével együtt) az ügynök nyújtja be, az ügynökazonosítót viszont a végrehajtó környezet szolgáltatja. (Az ügynök a saját azonosítójához természetesen nem férhet hozzá, mert ez hamisításra adna lehetőséget.) Ezeknek a jellemzőknek az ismeretében az ellenőrző algoritmus döntését a hozzáférést szabályozó adatbázis alapján hozza, és a végrehajtást szabályozó közeggel tudatja is. Ez utóbbi azután kötelező érvényre emeli azáltal, hogy a döntésnek megfelelően engedélyezi vagy megakadályozza a kezdeményezett művelet végrehajtását. Magától értetődőnek

tűnhet, de érdemes külön is hangsúlyozni: az ellenőrző algoritmus csak döntéseihez használhatja fel a hozzáférést szabályozó adatbázist, de nem módosíthatja és tartalmát a külvilággal nem ismertetheti meg.

Hozzáférést szabályozó adatbázis

Már az eddigiekből is következik, hogy a rendszer védelmét meghatározó adatokat mátrix formájában, az ún. **hozzáférési mátrixban** célszerű nyilvántartani. A hozzáférési mátrixban (8.3 ábra) minden egyes ügynökhöz egy sor, minden egyes adatelemhez egy oszlop tartozik, az $A_{(i,j)}$ elemben pedig az S_i ügynöknek az O_j adatelemmel kapcsolatos hozzáférési jogai (A_{ij}) találhatók meg. A hozzáférési jogokat az esetek nagy részében azoknak a műveleteknek a felsorolásával adjuk meg, amelyeket az ügynök a szóban forgó adatelemre alkalmazhat. Egy adatokat és programokat tároló adatbázisban például **adatelemeket olvashatunk** (r), **módosíthatunk** (w) és **törölhetünk** (d), **programok végrehajtását kezdeményezhetjük** (e), és **új adatelemeket hozhatunk létre** (a). Az egyes hozzáférési jogokat így egyszerűen az $r w d e a$ műveletek valamilyen kombinációja határozza meg. Amelyik ezek közül a hozzáférési jogban nem szerepel, annak végrehajtására az ügynök nem jogosult. A mátrix egy-egy sorát **hozzáférési tartománynak** nevezzük, mert a sorhoz tartozó ügynök által elérhető adatelemeket (engedélyezett felhasználási módjukkal együtt) tartalmazza. A mátrix mindegyik oszlopa egy-egy **hozzáférési lista**, amelyen az adatelemhez hozzáférő ügynökök (hozzáférési jogaikkal együtt) vannak felsorolva.

A hozzáférési mátrix az első pillanatban megtévesztően egyszerűnek tűnik. Egy működő rendszerben azonban dinamikus adatstruktúra: amikor ügynököt vagy adatelemet hozunk létre vagy meglévőket törölünk, a hozzáférési mátrix a formájában is válto-

	O_1	O_2		O_j		O_n
S_1						
S_2						
⋮						
S_i				A_{ij}		
⋮						
S_N						

8.3 ábra.

zik. Szövevényes problémahalmazra bukkanunk továbbá, ha a hozzáférési mátrix tartalmát vizsgáluk.

1. Elsősorban is a hozzáférési mátrix maga is védendő adathalmaz. Nem tekinthető azonban egyetlen adatelemnek, mert részei nem azonos védelmi igényűek. Ügynökök és adatelemek között például természetes módon létezik egy **tulajdonosi kapcsolat**, egy-egy ügynöknek a saját adatelemeire vonatkozó hozzáférési információk meghatározásában (tehát a megfelelő hozzáférési mátrixelem kitöltésében) megkülönböztetett joga van.

2. Új ügynök létrehozásának jogát sem gyakorolhatják a rendszer ügynökei egyformán, mert az új ügynök kezdeti jogainak helytelen beállítása az egész rendszer működésére potenciális veszélyt jelent.

3. A helyzetet bonyolítja az is, hogy az ügynökök — pontosabban programkomponenseik — létrehozásuk előtt adatelemek formájában részei a rendszernek. A „paszszív” adatelemek védelmi stratégiája tehát összefonódik az ügynökök létrehozását és működését szabályozó védelmi stratégiával.

4. Az ügynökök egy része maga is részt vesz a védelmi stratégia kialakításában, egyrészt azzal, hogy viselkedése „elvszerű” és „fegyelmezett”: a korlátozó végrehajtó közeggel szembeni „engedelmessége” „önkéntes”, vagyis az ügynök programba beépített ellenőrzésen, illetve hozzáférési stratégián múlik; másrészt azzal, hogy feladatát a megbízó ügynökhöz és annak jogaihoz igazítva végzi. A védelmi stratégia hajlékonysága és hatékonysága nem képzelhető el ilyen részlegesen vagy teljesen megbízható ügynökök (illetve a programjukat tartalmazó adatelemek) jelenléte nélkül. Erre egyszerű példát egy előfizetői katalógus szolgáltat. Ebben egy tételhez csak a szóban forgó előfizető (illetve ügynöke) férhet hozzá. Formálisan ezt úgy érhetjük el, hogy minden katalógustételt külön adatelemnek tekintünk, és hozzáférési információval látunk el. Ez azonban gazdaságtalan lenne a kis adatelemméret miatt. A másik megoldást megbízható ügynök alkalmazása jelenti. Ez a teljes katalógushoz hozzáférhet, de a beépített algoritmus folytán csak a pillanatnyilag kiszolgált előfizetőhöz tartozó katalóguseleme(ke)t keresi elő és teszi hozzáférhetővé.

5. A hozzáférési jogok általában továbbadhatók, átruházhatók — még hozzá többszörösen is. Enélkül a védelmi rendszer használhatósága túlságosan beszűkülne. A jogátruházás és -visszavonás egyidejű szabályozásának igénye azonban újabb problémákat teremt. Alapvető követelmény például, hogy egy hozzáférési jog hatásköre átruházással ne bővíthessen. Ellenkező esetben összejátszó ügynökök együttesen kijátszhatnák a rájuk egyedenként fennálló korlátozásokat.

A felsoroltakból is kitűnik, hogy a védelmi problémák további rendszerezése szükséges. A teljes védelmi rendszer céljait a **védelmi stratégia** foglalja magában, ez azonban lebontható olyan feladatokat megvalósító részterületekre, amelyek egymásra csak jól szabályozott (kevés paraméterrel leírható) módon hatnak. Ennek érdekében az ügynököket és adatelemeket elkülönítve kezeljük annak ellenére, hogy az ügynökök program formájukban adatelemek, ezért létrehozásuk vagy megszüntetésük adatelemek kezelését igényli. Az ügynök programot tartalmazó adatelemekre tehát a védelmi stratégia adatelemhozzáférésre vonatkozó részében csak egyetlen művelet alkalmazható: végrehajthatók. Az ügynökök létrehozásának, viselkedésük (működésük) módosításának, kizárásának (törlésének) szabályrendszerével a védelmi stratégia egy másik részterülete foglalkozik. Ugyancsak nagymértékben zárt és önálló stratégiai terület a hozzáférési jogok meghatározásának, szétosztásának, továbbadásának, visszavonásának a rendszere.

A védelmi stratégia leírását és érvényre juttatását a **védelmi eszközök** teszik lehetővé. A védelmi eszközök a védelmi rendszer stratégiától független formai elemei. A védelmi rendszer számára mindegyikük egy-egy értelmezési lépést, döntéshozatali eljárásrészletet határoz meg. Védelmi eszköz pl. maga a hozzáférési mátrix is, és a következő „személytelen” döntési eljárást testesíti meg: válaszsd ki a hozzáférési kérelmet benyújtó ügynökhöz tartozó sor és az igényben megjelölt adatelemhez tartozó oszlop közös elemét; az ebben talált hozzáférési jogban ellenőrizd a kért művelet jelenlétét; ennek alapján dönts el a kérelem engedélyezhetőségét. Ez az általános döntési séma akkor ír le és juttat érvényre adott védelmi stratégiát, ha a hozzáférési mátrixot (a védelmi eszközt) a megfelelő információval kitöltjük. (A sorokhoz rendeljük a rendszer ügynökeit, az oszlopokhoz az adatelemeit, és kitöltjük a hozzáférési jogokat.)

A hozzáférés-ellenőrzős tárgyalása során a továbbiakban kizárólag a védelmi eszközökkel foglalkozunk:

a) A védelmi stratégia igen szorosan kapcsolódik a mindenkorin rendszerhez, annak védelmi igényeihez, jellegzetességeihez, ezért egyöntetű kezelést biztosító modellünk mind ez ideig nincs rá.

b) A számos védelmi stratégiában hasznosítható — tehát általánosítható — mozzanatok automatikusan védelmi eszközzé válnak (annak definíciója értelmében).

8.2 A hozzáférési mátrix tárolása

A hozzáférési mátrix a védelmi rendszer központi eleme, első pillantásra tehát központi helyen való tárolása látszik nyilvánvaló megoldásnak. Ennek azonban olyan hátrányai vannak, hogy a hozzáférési mátrix ez esetben „messze kerül” mind az ügynököket, mind az adatelemeket gondozó rendszer részétől.

1. A hozzáférési mátrix dinamikus adatstruktúra, tárolókapacitás-igénye a rendszerbeli ügynökök és adatelemek számával változik. Központi tárolás esetén a hozzáférési mátrix elhelyezési problémáit külön kell megoldani az ügynök- és adatelem-gondozás hasonló problémáitól.

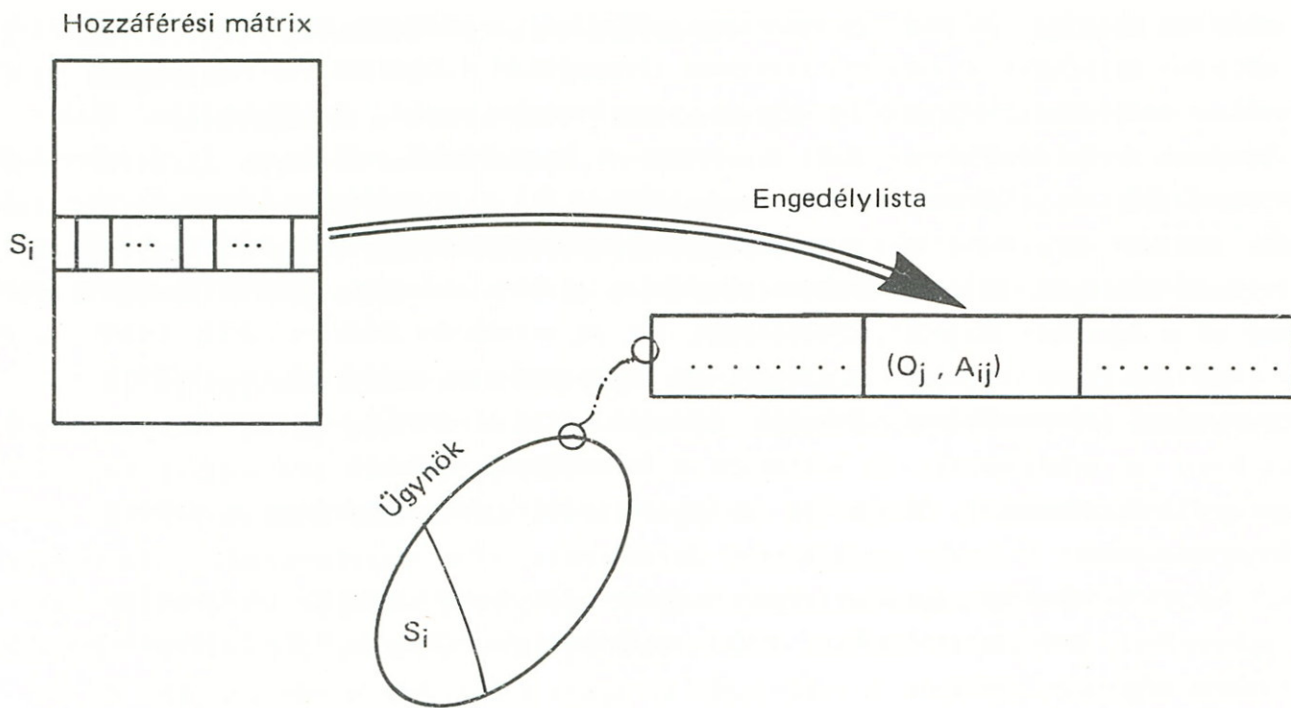
2. A hozzáférési mátrix egy-egy eleméhez vezető hozzáférési út is elkülönül a megfelelő adatelemhez vezető hozzáférési úttól, a szóban forgó ügynök rendszerbeli elhelyezkedését leíró információtól. Ez azt jelenti, hogy egy ügyletben (tranzakcióban) részt vevő ügynök és adatelem azonosítása után még egy további hozzáférési utat is végig kell járni. Ez elsősorban a rendszer robusztusságát csökkenti, mert rendellenes rendszerműködés esetén az újraindítási állapot visszaállításában nehézségeket okoz.

A hozzáférési mátrix tárolására éppen ezért olyan megoldást keresünk, amely szorosabban kötődik az ügynökök és adatelemek rendszerbeni megvalósításához.

8.2.1 Engedélyek

Egyik megoldásként a hozzáférési mátrixot tárolhatjuk az ügynökökhöz kötötten (8.4 ábra). Ebben az esetben a hozzáférési mátrix egy-egy sorát a megfelelő ügynökhöz, illetve a rendszerben őt leíró adatokhoz kapcsoljuk. Ennek formája egy olyan vektor, amelynek elemei

(adatelem, hozzáférési jog)



8.4 ábra.

párok, ún. **engedélyek**. Az S_i ügynök esetében a párok (O_j, A_{ij}) alakúak. A hozzáférést az ügynök a megfelelő engedély benyújtásával igényli. Az engedélyek természetesen nem hamisíthatók, az ügynök csak a rendszervédelem felügyelete alatt juthat birtokukba. Valójában az engedély tartalmát nem is kell ismernie, elegendő csupán azt tudnia, hogy bizonyos adatokat mely engedélyek segítségével érhet el.

Az engedély tehát kettős feladatot lát el:

- a) megnevezi az adatelemet, ezzel kijelöli a hozzáférési utat, és
- b) rögzíti az engedéllyel sikeresen kezdeményezhető hozzáférési műveleteket.

Az engedélyeket tárolhatjuk erre a célra külön kialakított adatstruktúrában (mint pl. az ábrán is tettük), ezt **engedélylistának** nevezzük. Az ügynök az engedélyre a listán belüli sorszámmal hivatkozik. Az engedélylista védett adatstruktúra, még az érintett ügynök sem férhet hozzá.

Ha magukat az egyedi engedélyeket védeni tudjuk, megengedhetjük, hogy az ügynökök a birtokukban lévő engedélyeket saját adataik között szabadon tárolják, és mozgassák. Ez az engedélyek használatát hajlékonyabbá teszi, a gyakorlatban azonban csak úgy oldható meg, ha az engedélyt tartalmazó tárolóelemek megjelölhetők, és ezzel a közönséges utasítások számára hozzáférhetlenné tehetők. (Hagyományos társzervezés mellett ez tárolóelemenként — pl.: szavanként — egy további bitet igényel, és így jelentős tárhelykapacitás-növekedést vonhat maga után.)

Engedélyek használata esetén a hozzáférési igény jogosságának ellenőrzése gyors és egyszerű: az engedély elővétele legfeljebb egy vektorelem kiválasztását igényli. Ugyan csak egyszerű az engedélyek szétosztása és továbbadása is. Az engedélyek visszavonása vagy a kiadott engedélyekben a hozzáférési jogok módosítása azonban már nehézségeket támaszt, mivel az engedélyek mozgása külön intézkedés nélkül követhetetlen. Az egyszer kiadott engedély például a továbbadás lehetősége miatt számos példányban létezhet, és ezeket egyszerre vagy valamilyen stratégia szerint kell visszavonni, módosítani. Megfelelő nyilvántartás hiányában az egyetlen megoldás egy globális engedélykeresés, amely azonban időigényes és a gyakorlatban ezért sokszor kivitelezhetetlen. En-

gedély alapú rendszerekben ezért az engedély-visszavonás problémája mindig jelen van, és olyan megoldást kíván, amely az engedélyek mozgását nyomon követi, és az erre a célra szolgáló adatstruktúrában rögzíti. Ezzel a feladatkörrel a későbbiekben még foglalkozunk.

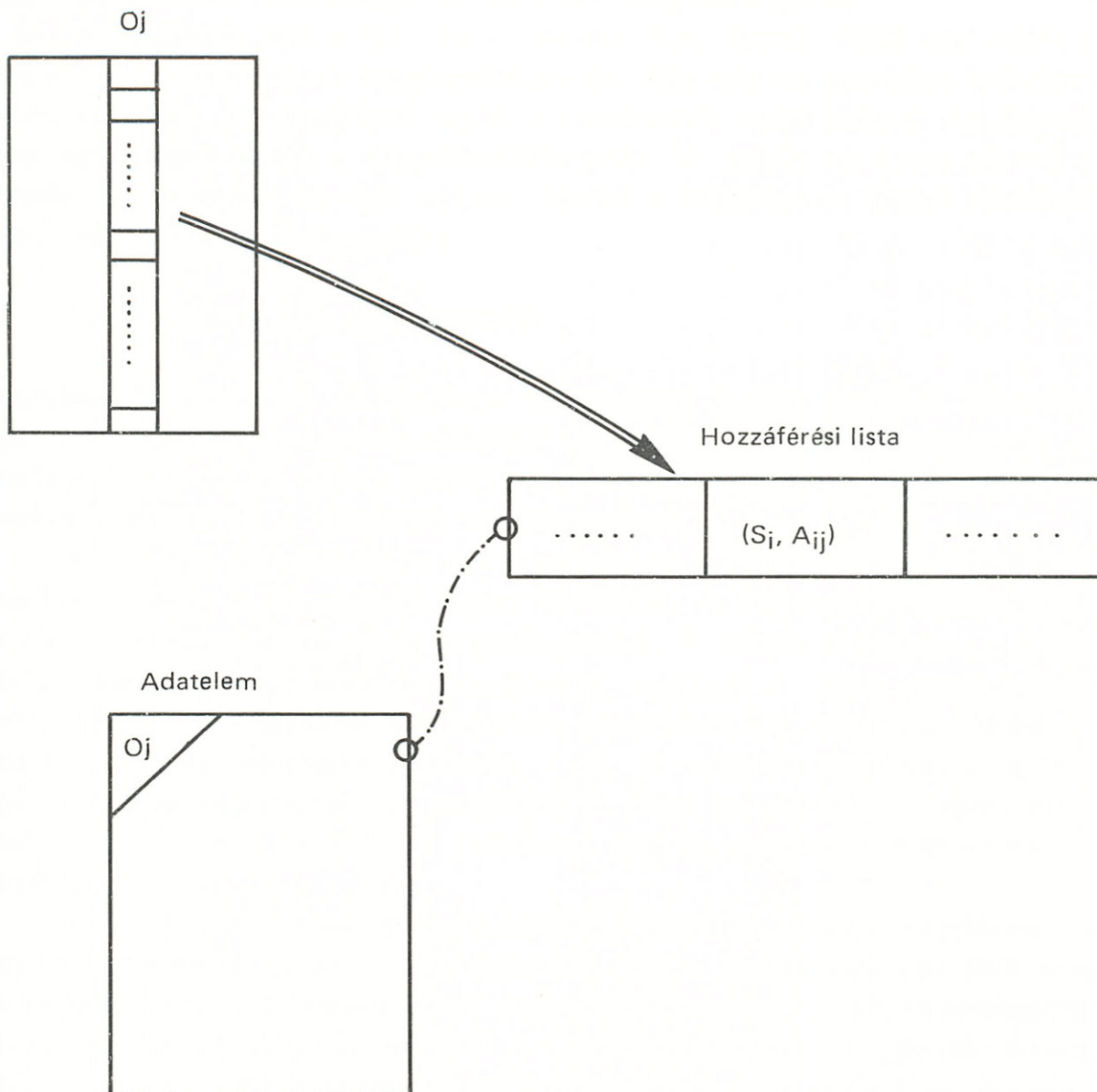
8.2.2 Felhatalmazások

A hozzáférési mátrixot tárolhatjuk az adatelemekhez kötötten is. Az eredmény (8.5 ábra) **hozzáférési listák** rendszere, amelyek mindegyike

(ügynök, hozzáférési jogok)

párokat, ún. **felhatalmazásokat** tartalmaz. Az O_j adatelemhez tartozó hozzáférési listán a felhatalmazások (S_i, A_{ij}) alakúak. A hozzáférési igény jogosságát a hozzáférési lista végignézésével lehet ellenőrizni. A hozzáférés engedélyezhető, ha a listán a megfelelő

Hozzáférési mátrix



8.5 ábra.

hozzáférési jogokat tartalmazó felhatalmazás szerepel. A hozzáférési listák használatának, mint látható, két jellegzetessége van:

- a hozzáférési jogok tárolása adatelemenként központosított, ezért az egy-egy ügy-
nökre vonatkozó módosításuk, visszavonásuk egyszerűen megoldható.
- az ellenőrzés időigényes művelet, mert a döntés alapját képező felhatalmazáshoz
csak kereséssel juthatunk el. A keresés időtartama a lista hosszától függ, ezért a
hozzáférési listák átlagos hosszának minél alacsonyabb értéken tartása elsődlege-
sen fontos feladat. A továbbiakban ennek megvalósításával is foglalkozunk.

9. A hozzáférés – védelem gyakorlati formái (példái)

9.1 Virtuális tár

A teljesség a legfontosabb követelmény, amelyet minden védelmi rendszernek ki kell elégítenie: a védett rendszerhez való minden hozzáférés ellenőrzött, a védelmi rendszer nem kerülhető meg. Számítógépes rendszerekben ennek az elvnek az érvényesítéséhez feldolgozás alatt a hozzáférést gépi utasításonként (pontosabban utasítás-elővételenként és operanduskidolgozásonként) újra és újra ellenőrizni kell. Mivel az ellenőrzés az utasítás végrehajtási idejét növeli, csak engedély alapú ellenőrzés használható, és az is csak megfelelő hardver eszközökre támaszkodó megvalósításban.

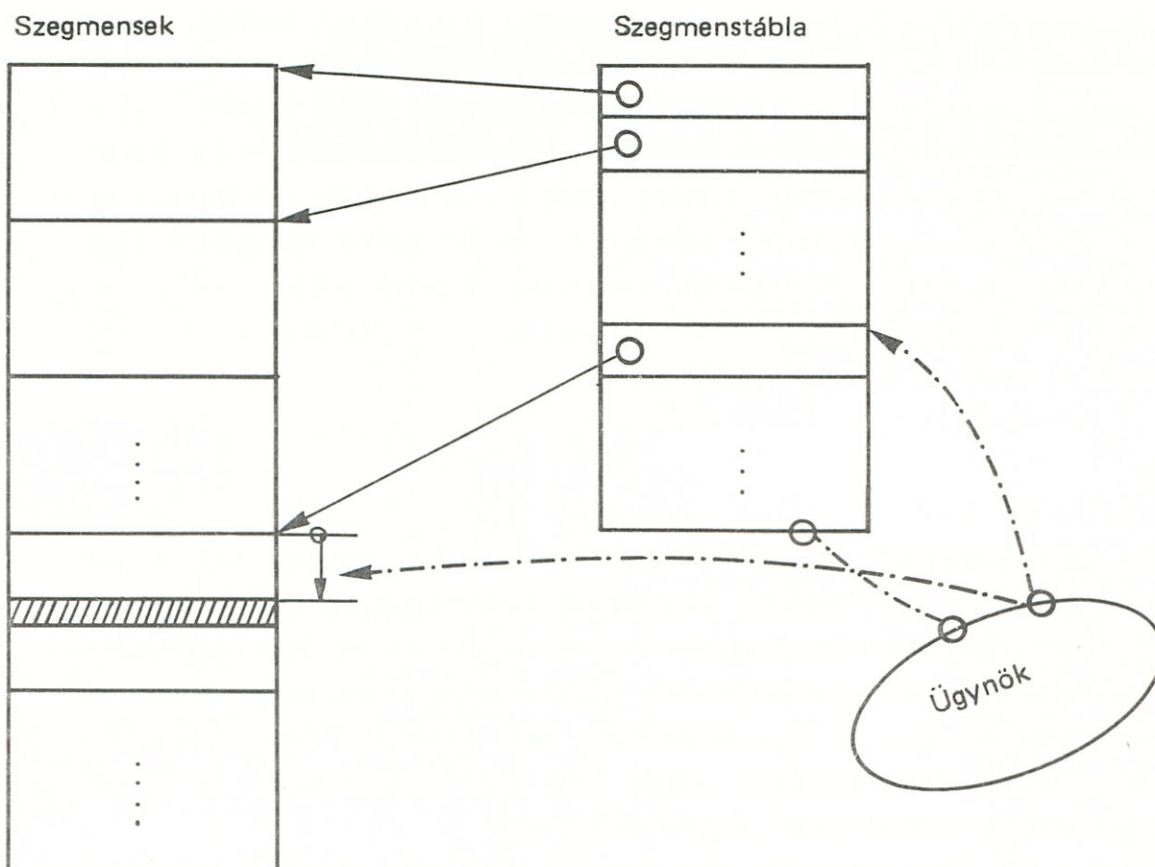
A megoldás elve a 9.1 ábrán látható. Egy ügynök által egy adott pillanatban elérhető teljes adathalmaz **szegmensekből** áll. Egy-egy szegmensben minden adat azonos védelmet élvez, a hozzáférési jogok a szegmensre vonatkoznak. Az ügynök rendelkezésre álló szegmenseket a **szegmenstábla** jelöli ki, amelynek egyes elemei a **szegmensleírások**. Ez utóbbiak tárolják többek között a szegmensek tárbeli kezdőcímét. A gépi utasításokban az operandusokra

(szegmenssorszám, szegmenscím)

formában hivatkozunk. A tényleges elérés tehát a szegmenstábla közvetítésével történik: első lépésben a szegmenstáblából elővesszük a szegmenssorszámnak megfelelő szegmensleírást, majd az abban fellelhető tárcímhez hozzáadva a szegmens relatív címet, kapjuk eredményül a tényleges operanduscímet.

Mivel egy szegmensleírás a szegmensre vonatkozó minden hozzáférés „útjába esik” megfelelő hely a hozzáférési jogok tárolására. A szegmensleírások tehát engedélyekké válnak, a szegmenstábla pedig engedélylistává. A szegmensleírásban a különböző hozzáférési jogokat egyedi bitek képvisel(het)ik, amelyek 1 értéke engedélyezést, 0 értéke tiltást jelent. A leggyakrabban használt hozzáférési jogok: **olvasás (read – r,)** **írás (write – w)** és **végrehajtás (execute – e)**. A számítógép központi egysége minden egyes utasítás végrehajtásakor megállapítja, hogy az utasítás az adott szegmensre vonatkozó melyik elemi hozzáférési műveletet foglalja magában, majd ellenőrzi annak jogoságát is.

A hozzáférés sebessége szempontjából lényeges, hogy a szegmensleírás elővétele a lehető legkevesebb járulékos időtöbblettel járjon. A használatban lévő szegmensleírásokat ezért gyors hozzáférésű belső regiszterekben, az ún. **szegmens-bázisregiszterekben** szokás tárolni. A szegmenstáblából a szegmensleírásokat az ügynök aktiválásakor a védelmi rendszer tölti a szegmens-bázisregiszterekbe, amelyekhez az ügynök közvetlenül

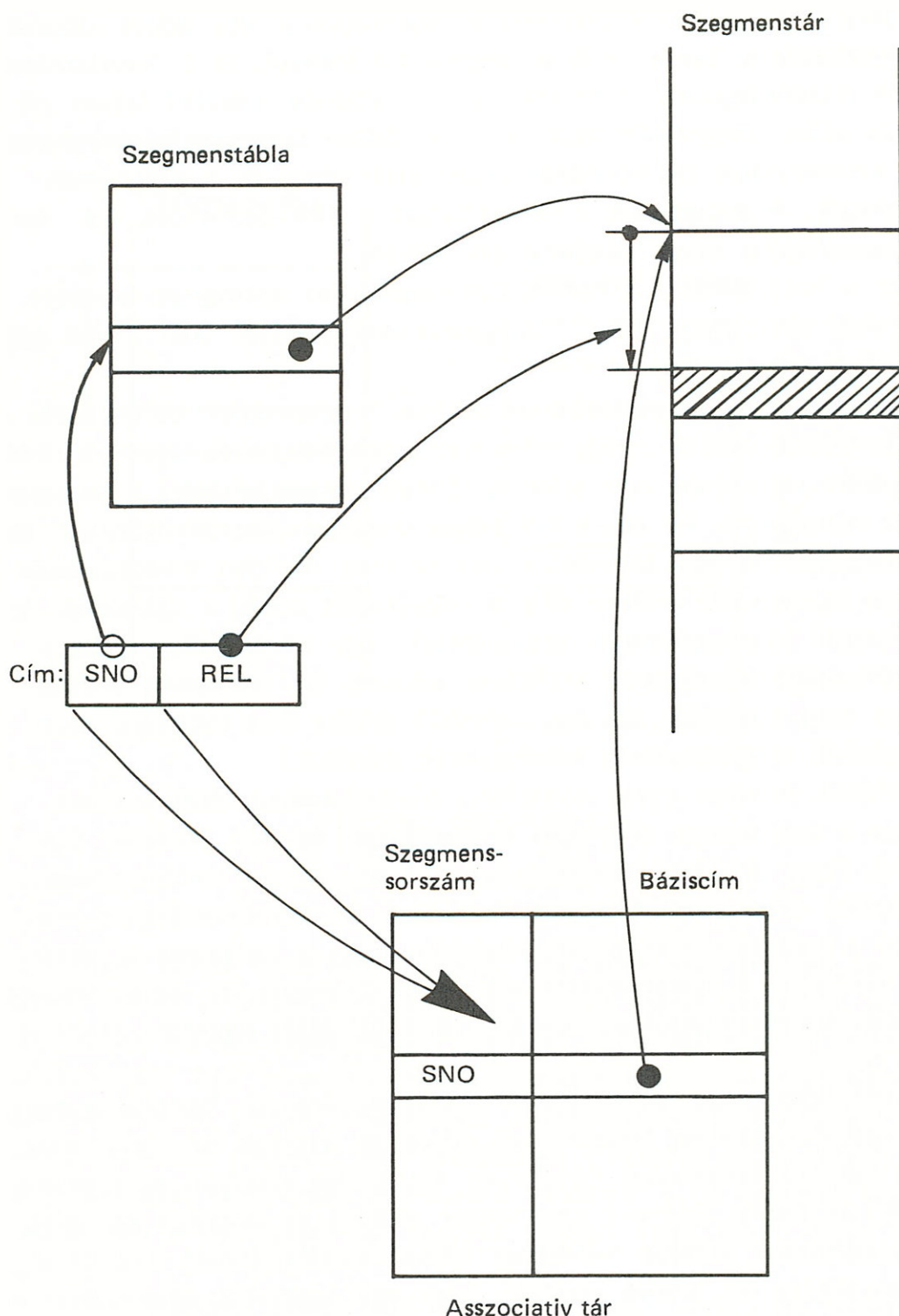


9.1 ábra.

nem férhet hozzá. Ugyancsak elérhetetlen az ügynök számára a szezmენტstábla is, pl. azért, mert a szezmensekhez, amely tárolja, szezmensleírással csak a védelmi rendszer rendelkezik.

Az egy ügynök által egyidejűleg használt szezmensek potenciálisan nagy száma miatt a teljes szezmენტstábla szezmens-bázisregiszterekben tárolása gazdaságtalan. A gyakorlati megoldások mindegyikében csak az éppen használatban lévő szezmensleírásokat (szám szerint 8–16–32) tároljuk regiszterekben. Egy szezmensleírást a szezmენტstáblából akkor töltünk szezmens-bázisregiszterbe, amikor a szóban forgó szezmens használata szükségessé válik. A szezmensleírások mozgatása történhet „manuálisan”: az ügynök program kezdeményezi és tartja nyilván a szezmensleírások és szezmens-bázisregiszterek összerendelését. Az ügynök program tehát szezmens-bázisregiszterekre és nem szezmensekre hivatkozik. A kívánt szezmens elérését az biztosítja, hogy a megfelelő szezmens-bázisregiszterbe az ügynök program betölteti a szezmensleírást a védelmi rendszerrel (mivel a szezmens-táblához és a szezmens-bázisregiszterekhez közvetlenül nem fér hozzá). Ezt a megoldást számos forgalomban lévő számítógéptípusban megtalálhatjuk, legismertebb példáját a Digital Equipment Corporation (DEC) PDP–11 családja szolgáltatja.

Egyre inkább terjed azonban a szezmens-bázisregiszterek kiosztásának automatikus megoldása. Alapelvét a 9.2 ábra szemlélteti. A szezmens-bázisregisztereket kiegészítettük egy további mezővel, amely az illető szezmensleírás szezmენტstáblabeli helyét, **szezmens-sorszámát** tartalmazza. Ezek a mezők együttesen asszociatív, tehát tartalom szerint címezhető tárat alkotnak. A szezmensekre sorszámukkal (szezmენტstáblabeli indexükkel) hivatkozunk, amellyel a szezmens-bázisregiszterekben asszociatív (párhuzamos) keresést kezdeményezünk. Ha ez sikeres, egyetlen regiszter-hozzáférési ciklusban a megfelelő szezmens-bázisregiszter kiválasztása és tartalmának, a keresett szezmensleírásnak a kiolvasása megtörténik. Ha a keresett sorszámú szezmensleírás nincs egyik szezmens-bázisregisz-



9.2 ábra.

terben sem, a ciklus végén negatív jelzés keletkezik, amely a szegeztáblabeli szegeztábla leírás elővételét kezdeményezi. Ennek során első lépésben egy fogadó regisztert választunk ki. Ha van(nak) üres regiszter(ek), akkor ez(ek egyike) lesz az, ha nincs(enek) akkor a helyettesítő algoritmus választ egyet a foglalt regiszterek közül. Második lépésben a keresett szegeztábla leírást a kiválasztott szegeztábla-bázisregiszterbe töltjük (ennek korábbi tartalmát felülírva). Egyidejűleg a szegeztábla leírás sorszámát a regiszter asszociatív mezejébe írjuk.

A megoldás határfokát nyilván az jellemzi, hogy a kezdeményezett hozzáférések mekkora hányadában zárja rövidre az asszociatív keresés az eredeti szegeztábla igénybevételét. A határfokot viszont három tényező határozza meg:

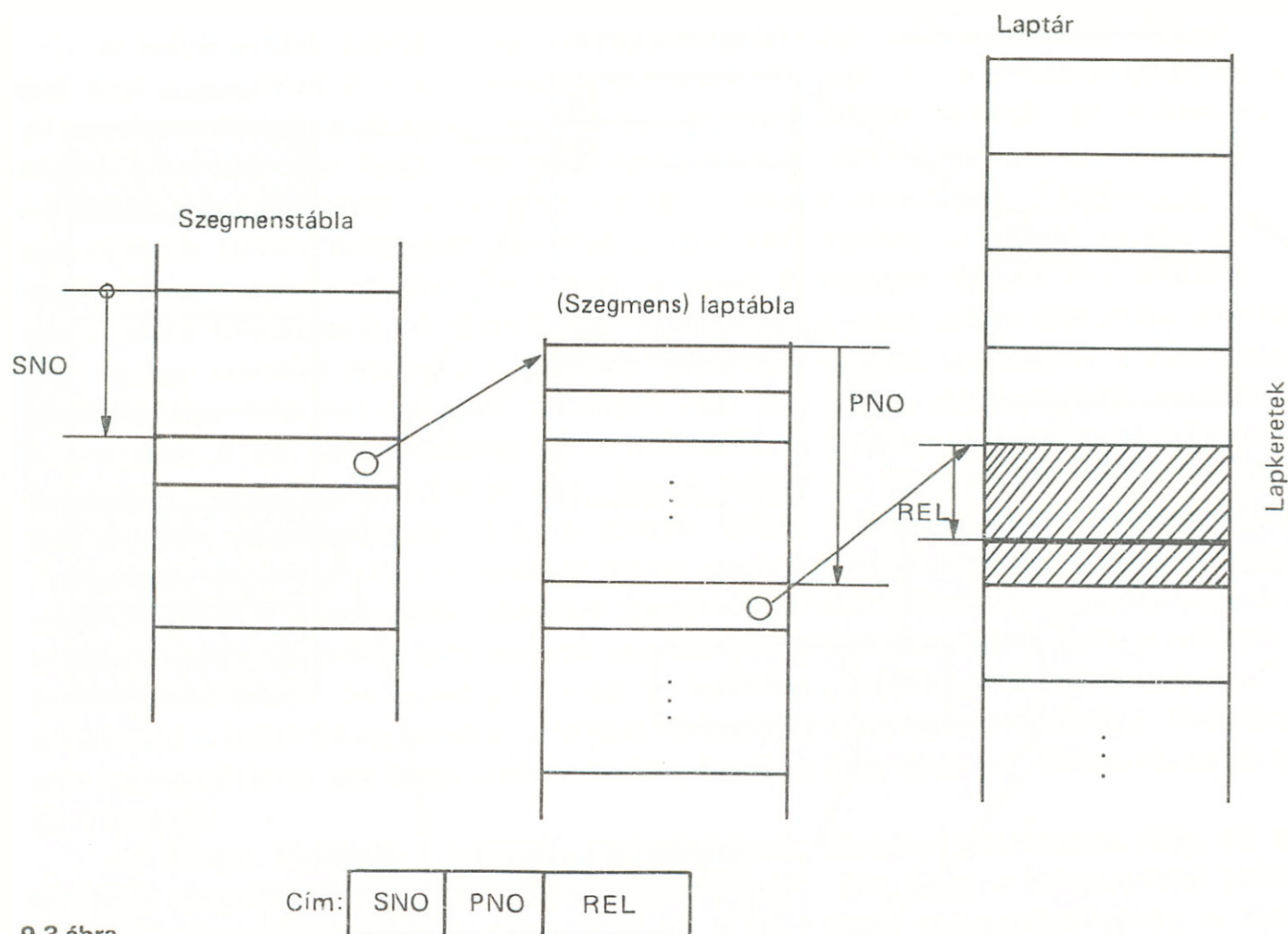
- Alapvető szerepet játszik a szegmenshivatkozások csomósodása: egy adott időpont környezetében a hivatkozások zöme néhány szegmensre irányul, és a hivatkozásoknak e magja időben viszonylagosan (a hivatkozások számához mérten) lassan változik. (A mindenkor aktív szegmensleírások tehát kisszámú szegmens-bázisregiszterben tarthatók.) A hivatkozások csomósodása tapasztalati tény, és a programok szerkezetéből következik. A programok automatikusan is átrendezhetőek (pl.: fordításkor), hogy a csomósodás minél tökéletesebb legyen.
- Nyilvánvalóan növeli a határfokot a szegmens-bázisregiszterek számának növelése. Ismert csak tapasztalati tény, hogy 16–32 szegmens-bázisregiszter már kiváló eredményeket tesz lehetővé.
- Végül, de nem utolsósorban, a helyettesítő algoritmus is alapvetően befolyásolja a rendszer működését. Nyilvánvaló cél, hogy mindig az a szegmensleírás legyen felülírva, amelyre a közeljövőben legkevésbé lesz szükség. Ennek megvalósítására a szegmensleírások fontosságát mérni kell. Az egyik lehetséges stratégia megvalósításához minden szegmens-bázisregiszterhez egy számlálóra van szükség. Minden hivatkozáskor a számláló értékét egy legnagyobb értékre állítjuk, egyébként pedig a számlálók értékét állandó sebességgel (párhuzamosan) csökkentjük. Egy adott időpontban a szegmensleírások fontossága számlálójuk értékével arányos. (A számlálók értékét negatív tartományba nem engedjük. Ha egy számláló értéke 0-ra csökken, azon is marad, és a megfelelő szegmensleírás feleslegesnek minősül.)

Mivel a szegmensleírások minden egyes hozzáférés kialakításában részt vesznek, felhasználhatók arra, hogy a szegmensek tényleges elérhetőségét bennük jelöljük. Egy további bitben jelezhetjük, hogy az illető szegmens az operatív tárban tartózkodik-e. Ha ennek a bitnek az értéke (mondjuk) 0, akkor hozzáférés kezdeményezésekor megszakítás keletkezik, amely a működtető rendszert aktiválja, hogy a szegmens a háttértárról az operatív tárba kerüljön. A szegmensekre alapozott címmező így válik virtuális tárrá: a program-végrehajtás szempontjából az operatív és a háttértár egyetlen tárolóvá olvad össze.

A szegmensek változó hosszúságú tartományok (címtartományok), bennük a megoldandó feladat szempontjából logikailag összetartozó adatokat tároljuk (pl.: egy programmodul utasításait). A változó hosszúságú tárigények kielégítése azonban az operatív tár feldarabolásához vezet: az éppen rezidens szegmensek mellett az operatív tár jelentős kapacitáshányada a szegmensek átlagos méretéhez képest kicsiny (tehát használhatatlan) izolált részletekben lesz jelen. Ennek oka az, hogy egy szegmens elhelyezésekor általában nem rendelkezünk pontosan a szegmensméretnek megfelelő szabad tárrészlettel, azt egy nagyobb méretűből kell kihasítanunk. A maradékok egy idő után a legkondosabb gazdálkodás mellett is használhatatlanul kis méretűek lesznek. (Hallgatólagosan feltételeztük, hogy az egyes betöltött szegmensek használatuk alatt az operatív tárban nem mozgathatók.)

Az operatív tár kihasználatlanságát növeli az a tény is, hogy a lokalitás a szegmenseken belül is érvényes: egy programvezérlés vagy adathozzáférés hivatkozásai egy-egy rövid időintervallumban a szegmens teljes címtartományának egy kis hányadát kitevő címekre vonatkoznak, és ezek a címcsomók méretükben és helyükben viszonylagosan lassan változnak. Végrehajtás közben egy-egy szegmens jelentős része tehát „holt teher”, feleslegesen foglal le operatív tárat.

Az operatív tár kihasználatlanságának ezt a problémáját megoldhatjuk a szegmens-

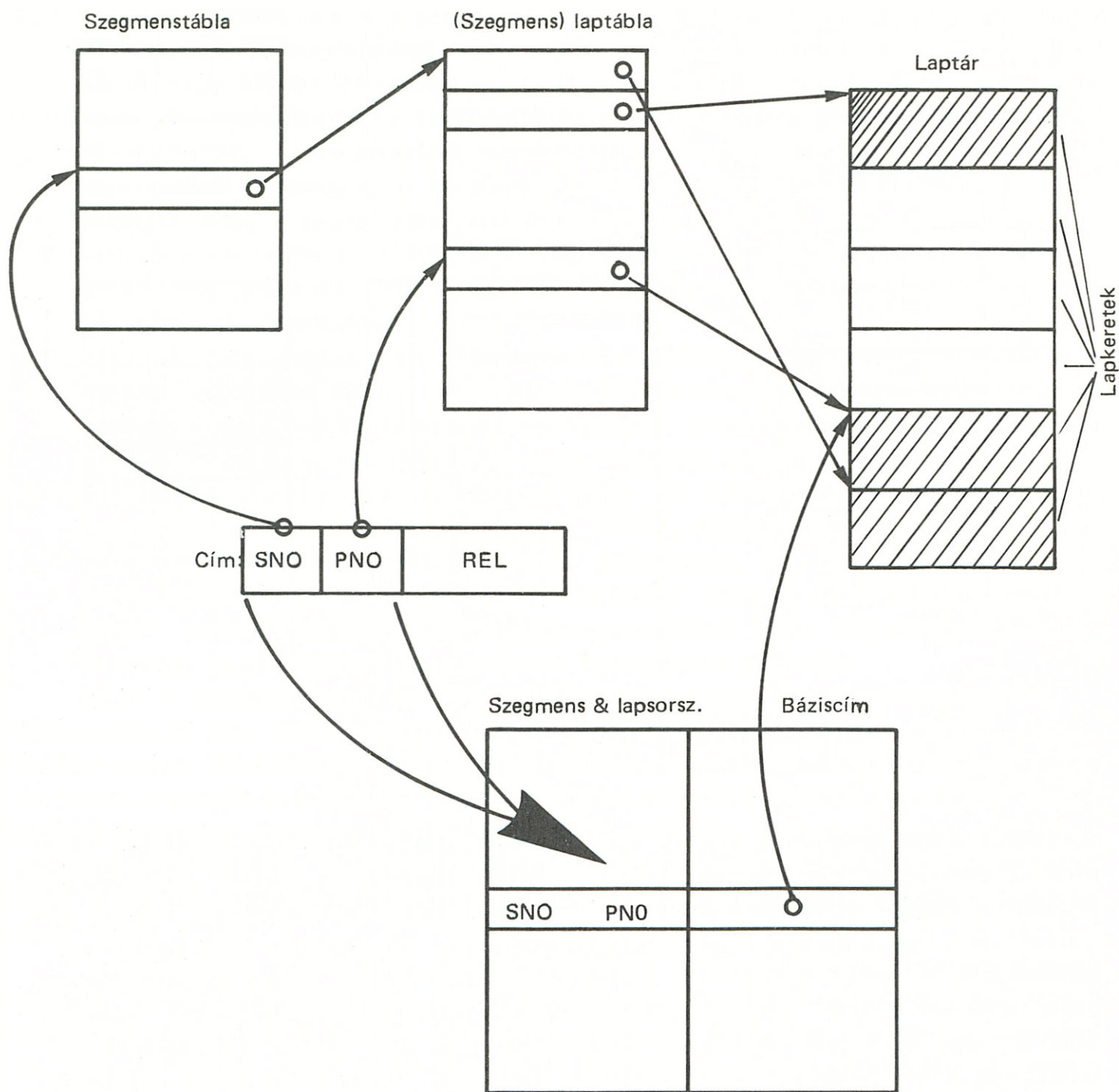


9.3 ábra.

hozzáférés előbb ismertetett elvének egy következő szinten való megismétlésével (9.3 ábra). Ehhez az egyes szegmenseket azonos méretű **lapokra**, az operatív tárat pedig ezzel azonos méretű **lapkeretekre** bontjuk. (Egy lapkeretben tehát pontosan egy lap helyezhető el.) Egy szegmens elhelyezkedését most már nem egyetlen cím, hanem egy **laptábla** írja le. Ennek megfelelően a szegmensleírás laptáblára mutat, a szegmenshossz pedig lapokban értendő. Egy tetszőleges szegmens egész számú lapokból áll, legutolsó lapja azonban lehet csak részlegesen kitöltött is. A laptábla elemei, a **lapleírások** mutatják meg a lap helyét az operatív tárban, vagy jelezzik, hogy a lap nincs oda betöltve.

Az operatív tárhoz való hozzáférés most kétlépcsős, először a hivatkozás szegmens részét felhasználva a szegmenstáblához fordulunk. Ezután a szegmens relatív címéből meghatározzuk a lap sorszámát és a lap relatív címét. Az előbbivel érjük el a megfelelő lapleírást, illetve a lapot tartalmazó lapkeretet, az utóbbival pedig ezen belül a keresett tárolóelemet. Hiányzó lapok lapkeretbe töltése itt is automatikus. A 9.4 ábra szerint ebben az esetben is alkalmazhatunk asszociatív tárat, ez azonban most (szegmenssorszám, lapsorszám) párokat tartalmaz, és a párhuzamos keresés ezekre vonatkozik.

Mivel a lapokra bontás célja a szükséges operatív tár-méret csökkentése, a lapok és lapkeretek összerendelése helyettesítési problémát jelent: valahányszor egy új igény keletkezésekor nincs szabad lapkeret, egy operatív tárbeli lapot át kell írni. Az átírandó lapot természetesen úgy kell kiválasztani, hogy az „okozott kár” minimális legyen. A helyzetet bonyolítja, hogy az operatív tárban egyidejűleg több konkurrens folyamat adat- és programkomponensei találhatóak.



9.4 ábra.

Asszociatív tár

Egy lehetséges (és egyben a jelenleg ismert leghatásosabb) stratégia a következő:

a) Meghatározzuk egy folyamat átlagos operatív-tár-igényét és azt fenntartjuk számára. (A tárigény az időben változhat, folyamatos meghatározásának módját a továbbiakban ismertetjük.)

b) Egy folyamat élete során laphelyettesítési igényeit csak saját operatív-tár-kapacitásából elégítjük ki. („Kannibalizmust” tehát nem engedünk meg, egy folyamat mások rovására csak a fent említett átlagos tárkapacitásméret-növekedés következtében terjeszkedhet.)

c) A rendszer csak annyi aktív folyamatot engedélyez, amennyinek az átlagos operatív-tár-igénye a fizikai tárból kielégíthető.

A helyettesítési döntés alapja a folyamat mindenkor **referenciabázisa**. Ezt a folyamat által a megelőző T s alatt megcímezett lapok alkotják. T megfelelő meghatározásával a referenciabázis csak lassan, mérete pedig még lassabban változik. (T értékét ellentmondó követelmények határozzák meg, amelyekkel itt nem foglalkozunk, értéke 10–200 ms között van, elsősorban a háttértár-hozzáférés átlagos sebességétől függően.) A folyamat operatív tárban tartózkodó lapjainak a referenciabázishoz tartozását egyszerűen mérhetjük. Minden laphoz számlálót rendelünk, amelyet T értékére töltünk fel, valahányszor a lapra hivatkozunk. A számlálót egyidejűleg az eltelt idővel arányosan csökkentjük. Ha egy számláló értéke 0-ra csökken, akkor a megfelelő lap kikerül a referenciabázisból. (Egy folyamat lapjainak számlálóit csak a folyamat aktív idejében működtetjük, és értéküket 0 alá nem csökkentjük.) A számlálók optimálisan a számítógép központi egységének részeként valósíthatók meg. Ennek hiányában azonban szoftverben és működtető rendszer szolgáltatásaként is kialakíthatók. Ehhez a központi egység sebességétől és T értékétől függően 1–10 ms ismétlési idővel megszakításokat keltünk, és ezek kiszolgálásakor végezzük el a számlálók értékének beállítását. Ehhez segítséget nyújt az, hogy a lapleírások egyéb célokból is tartalmazzanak egy hozzáférés bitet, amelyet a hardver minden egyes hozzáféréskor beállít. (A teljesség kedvéért említjük meg, hogy egy másik, ún. „piszkos” bit is található a lapleírásban, amelyet a hardver akkor állít, ha a lapba írás történt. Valahányszor helyettesítéskor egy lapot átírásra ítélünk, háttérre csak akkor kell kiírunk, ha ez a bit 1 értékű.)

Az átlagos tárigényt a laphibák gyakoriságának mérésével határozzuk meg. Ha ez egy előre megadott P küszöböt meghalad, az annak a jele, hogy a folyamathoz példánnyal hozzárendelt lapkeretek száma nem éri el az átlagos tárkapacitás-igényt. A folyamat új lapigényét ezért a közös tárkapacitásból elégítjük ki, esetleg más folyamat rovására. Ha a laphiba-gyakoriság kisebb az előbbi küszöbnél, akkor viszont a folyamat lapkereteinek száma csökkenthető más folyamatok javára.

Végül megjegyezzük, hogy a lapokra bontásnak a rendszervédelemben is van szerepe. Amikor egy folyamat B/K (beviteli + kiviteli) műveletet kezdeményez, tevékenysége (előbb-utóbb) felfüggesztődik, és az operatív tárban csak a mozgatott adatokat tároló tartományra van szükség. Ennek egysége célszerűen a lap, és a bevitel-kivitel alatt elegendő csak a lapokat védeni. A szokásos megoldás minden laphoz egy kulcs hozzárendelését teszi lehetővé (általában 4–6 bit hosszúságú). A lap kulcsát a B/K vezérlőegység is megkapja, számára a laphoz férést saját és az etérendő lap kulcsának egyezése engedélyezi. Kulcseltérés esetén a hozzáférést az ellenőrző hardver letiltja. A 0 kulcs általában kitüntetett szerepet játszik, összehasonlításakor bármelyik kulccsal megegyezik. Ez teszi lehetővé közös lapok használatát, illetve a felügyelőprogram számára a tetszőleges laphoz való hozzáférést kulcsuk értékétől függetlenül. Lapkulcsok használata védelmet nyújt az ellen, hogy külső folyamatok (mint pl.: B/K) hibás működés esetén kiterjedt mértékű károkat okozzanak.

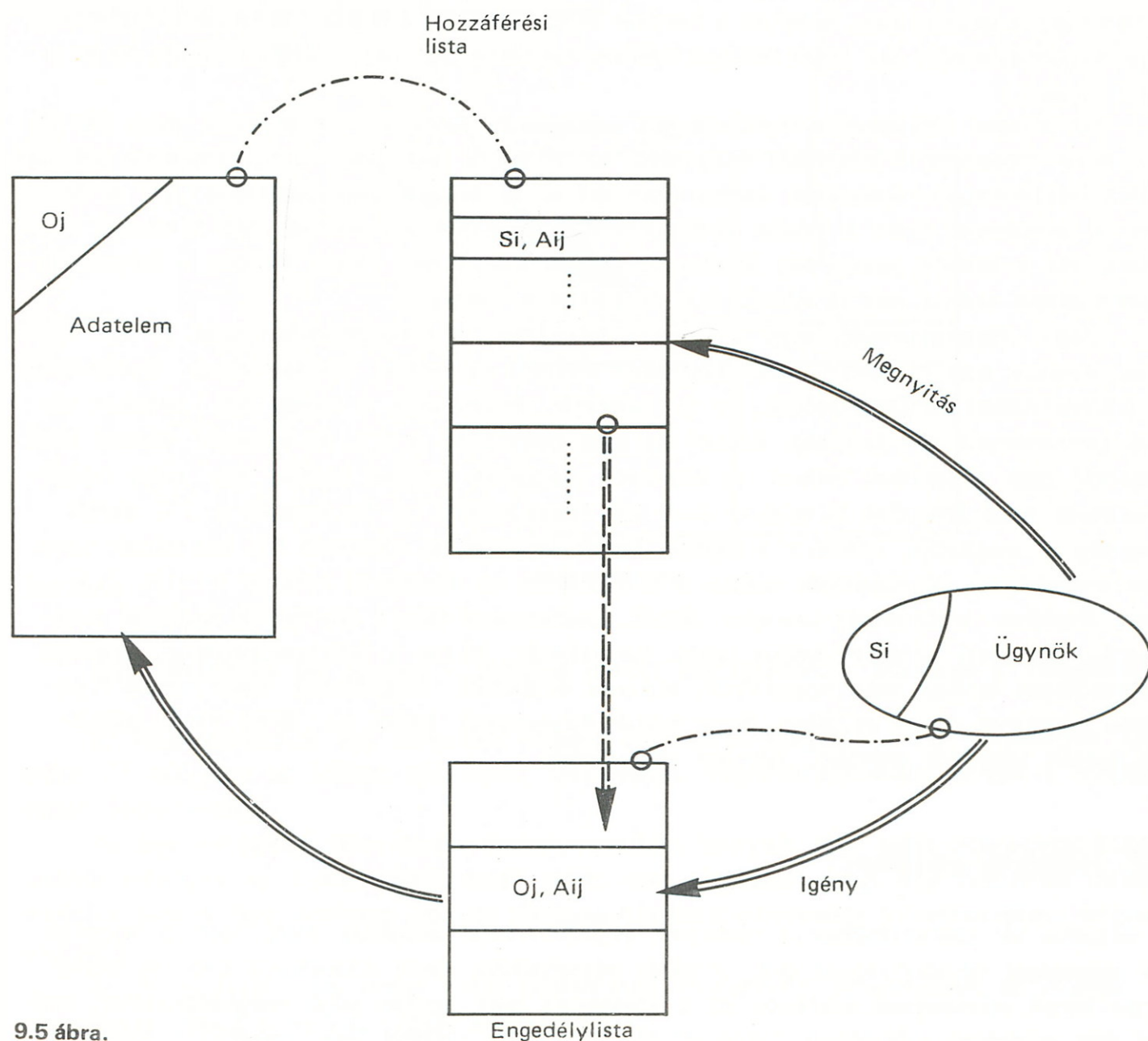
9.2 Háttértár-védelem

A virtuális tár szervezésében a védelmet engedélyekkel oldottuk meg, mert a hozzáférés jogosságát minden egyes gépi utasítás végrehajtása során ellenőrizni kell. A háttértáron tárolt adatalemek esetében ez a szempont csak azután válik meghatározóvá, amikor egy adatalem valamelyik folyamat címmezejének részévé lesz (bekerül a folyamat

virtuális tárjába). Ezeknél az adatelemeknél viszont a védelmi információ szoros felügyelete és így térbeli koncentrációja fontos. (Háttértáron tartott adatelemekeken keresztül valósul meg a rendszer közös adatainak használata, illetve az egyes felhasználók közötti adat- és információcsere.)

A háttértári adatelemek védelmének elsődleges módja ezért a hozzáférési lista, a védelem teljes szervezése pedig szorosan kapcsolódik az ilyen adatelemek hozzáférési módjához (9.5 ábra).

Az adatelem-hozzáférés megnyitásakor (OPEN) az adatelemhez tartozó hozzáférési lista szolgál az engedélyezés alapjául. Ha a kezdeményező S_i ügynököt tartalmazó (S_i, A_{ij}) felhatalmazás a hozzáférési listán szerepel, abból a rendszer egy (O_j, A_{ij}) engedélyt állít elő, és azt az ügynök engedélylistájához csatolja. A további hozzáférési igények benyújtása, ellenőrzése és teljesítése már az engedélyek felhasználásával történik. Az ügynök elől ezt a változást a legtöbb modern működtető rendszer eltakarja. A háttértári adatelemekhez adatelemenként, egy-egy vezérlőtáblázat segítségével lehet hozzáférni. Ez tartalmazza az adatelemre vonatkozó összes fizikai paramétert (többek között a megfelelő engedély listabeli indexét is), az ügynök igényeiben csak erre a vezérlőtáblázatra hivatkozik.



9.5 ábra.

9.3 A hozzáférési lista szerkezete

A hozzáférési listák használatának költsége átlagos hosszuktól függ: a tárolásukhoz szükséges tárkapacitás holt teher, a felhatalmazások kereséséhez szükséges végignézésük idővesztés. Mindkettőt egyidejűleg minimalizálhatjuk, ha a védelmi információt a lehető legkevesebb számú felhatalmazással fogalmazzuk meg.

A hozzáférési listáról az előzőekben meglehetősen egyszerű képet alkottunk: minden felhatalmazás egyetlen ügynökre vonatkozott, és ha egy ügynök egyetlen felhatalmazásban sem szerepelt, akkor a szóban forgó adatelemhez semmilyen hozzáférési joga sem volt. Ez utóbbi az ún. **zárt rendszerek** alapelveinek felel meg: egy adatelem mindaddig elérhetetlen, amíg arra kifejezetten engedélyt nem adunk. Zárt rendszerekkel hatékonyan modellezhetünk olyan rendszereket, amelyekre az egyes felhasználók elszigeteltsége jellemző, az adatelemek zömét csak tulajdonosa használja.

A zárt rendszerek nehézkessé válnak azonban, mihamarabb az adatelemek közös használata. az egyik meghatározó szempont a rendszer használatában. A fenti egyszerű hozzáférési lista-modellhez ugyanis adatelemenként minden hozzáférési joggal rendelkező ügynököt egyenként fel kell sorolni. Az eredmény: hosszú és így gazdaságtalan hozzáférési listák. Valójában ez a probléma minden rendszerben jelen van, mert még a legegyszerűbb rendszerek is tartalmaznak közös használatú, minden felhasználó (illetve ügynök) számára hozzáférhető adatelemeket.

Az orvoslás első lépése annak felismerésén alapul, hogy egy rendszer használói általában különböző csoportokhoz rendelhetők, és egy-egy csoport tagjait a rendszer használatában azonos törvények szabályozzák.

A hozzáférési listákon így elegendő csupán az egyes csoportokat szerepeltetni. A csoportokra bontás természetesen adódik.

a) Az egyedi felhasználókat ebben a megközelítésben egyelemű csoportoknak tekintjük.

b) Csoportokat alkothatunk a felhasználók minősítésével. Például:

- *Egyszerű felhasználó*: Általában csak használhatja a rendszer szolgáltatásait, de meg nem változtathatja.
- *Rendszergazda*: A rendszer felhasználását érintő kérdésekben dönthet (például új felhasználók felvétele a rendszerbe).
- *Rendszerszervező*: A rendszer szolgáltatásainak meghatározása a feladata.
- *Rendszerprogramozó*: A rendszer működését érintő kérdések tartoznak hozzá.

A rendszer egyes szolgáltatásait megvalósító programok hozzáférési listáin például elegendő lehet csupán a fenti minősítésekkel meghatározott csoportok azonosítóját feltüntetni az egyes felhatalmazásokban. Az ügynök hozzáférési jogának megállapítása most nem azonosítója alapján, hanem minősítése szerint történik.

c) Csoportokat alkothatunk a projektek vagy a számítógép-használatot finanszírozó költséghely szerint.

d) Csoportot határozhat meg egy adott alkalmazás is: például egy számítógéppel segített oktatási rendszerben egy tancsoport (tankör).

Mivel a csoportazonosítók részt vesznek a hozzáférési stratégia kialakításában, éppen úgy védelemre szorulnak a visszaélésekkel szemben, mint az egyes felhasználók számítógépen belüli azonosítója. A csoportok megalakítása, a csoportokhoz tagok hozzá-

rendelése, a csoportok tagjainak nyilvántartása stb. mind olyan tevékenységek, amelyeket a védelmi rendszerben szabályozni kell. (Csoportokat „alapító” felhasználót például csak rendszergazda nevezhet ki, egy felhasználó, illetve ügynöke valamilyen csoporthoz való tartozását éppen úgy nem hamisíthatja meg, mint rendszerbeli azonosítóját sem.)

A csoportok képzése önmagában azonban még nem biztosítja a hozzáférési listák gazdaságosságát. Ha például egy adatelemet egy már létrehozott csoport számára hozzáférhetővé teszünk, majd ezt a jogot a csoport egy (vagy néhány) tagjától vissza akarjuk vonni, két kevésbé vonzó lehetőséggel állunk szemben.

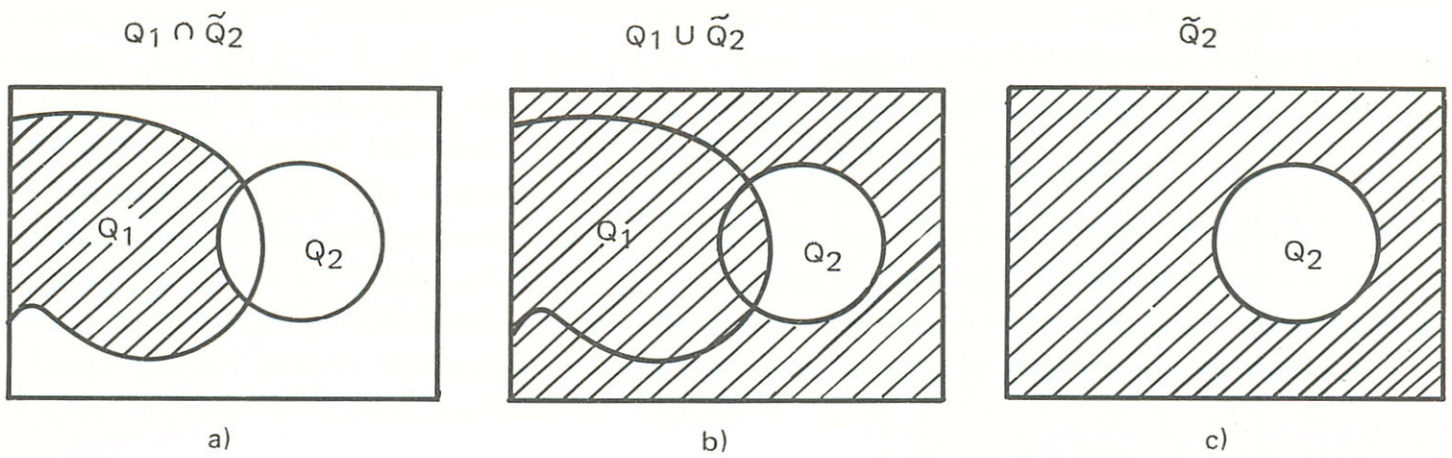
- Az engedéllyel továbbra is rendelkező ügynököket egyenként felsoroljuk a hozzáférési listán, és ezzel a csoportképzés előnyeit elveszítjük.
- Egy új csoportot hozunk létre, amelybe a kizárt ügynökök már nem tartoznak bele. Ez a megoldás két ok miatt sem elfogadható.
- Egyrészt a csoportok létrehozása és nyilvántartása költséges (idő- és szervezésigényes), ami korlátozza a rendszeren belül együtt létező csoportok számát, illetve összetételük változtatási gyakoriságát.
- Másrészt egy-egy ügynök előbb-utóbb olyan sok csoport tagja lesz, ami már szintén nyilvántartási és felhatalmazás-ellenőrzési problémákhoz vezet.

Az ebben a példában felmerülő feladatot úgy oldhatjuk meg, ha a felhatalmazások mellett bevezetjük a kizárások használatát is, amelyek megtiltják egy adatelem használatát (pontosabban azoknak a hozzáférési módoknak a használatát, amelyek a kizárásban szerepelnek). Mint ahogyan a felhatalmazások a zárt rendszerekben, a kizárások a **nyílt rendszerekben** módosítják a hozzáférés általános szabályát. Egy nyílt rendszer alap helyzetében egy adatelem mindaddig elérhető az ügynök számára, amíg azt kifejezetten nem tiltjuk. (Az üres hozzáférési lista tehát megszorítás nélküli elérhetőséget jelent.)

Engedélyező előírások (felhatalmazások) és **tiltó előírások** (kizárások) együttes használatakor adatelemenként dönthetjük el, hogy használata nyitott vagy zárt-e. Ha az „összes felhasználó” csoportja jelölhető, akkor ezzel a csoporttal egy felhatalmazást kezdő elemként elhelyezve a hozzáférési listán, az adatelem nyílt, kizárással kezdve a listát, az adatelem zárt. Az üres hozzáférési lista értelmezését a teljes rendszer nyíltsága vagy zártsága határozza meg, de ilyen alapértelmezés rögzítése nem kötelező. (Alapértelmezés hiányában természetesen az üres hozzáférési lista használata nem megengedett.)

Zárt hozzáférési listákon a felhatalmazások, nyílt hozzáférési listákon a kizárások használata magától értetődő. Ha azonban engedélyező és tiltó előírások együttesen szerepelnek, a hozzáférés elbírálásában értelmezési bizonytalanság keletkezik. Ha ugyanis az A ügynök a Q_1 és Q_2 csoportnak is tagja, hozzáférhet-e egy adatelemhez, ha annak hozzáférési listáján a Q_1 csoport felhatalmazásban, a Q_2 csoport pedig kizárásban szerepel?

- Ha a tiltó előírásnak elsőbbsége van az engedélyező előírással szemben, akkor az értelmezés: „az adatelemhez hozzáférhetnek a Q_1 csoport tagjai, kivéve azokat, akik egyben a Q_2 csoportnak is tagjai” — vagyis az A ügynök számára a hozzáférés nincs engedélyezve. Másképpen fogalmazva: a két előírás együttesen a $Q_1 \cap Q_2$ halmazba tartozó ügynököknek engedélyez hozzáférést (9.6 a) ábra). (A kiegészítő halmaz, Q_2 képzése az ügynökök teljes halmazához képest történik (9.6 c) ábra.)
- Ha viszont az engedélyező előírás az „erősebb”, akkor az értelmezés így módosul: „az adatelemhez nem férhetnek hozzá a Q_2 csoport tagjai, kivéve azt az esetet,



9.6 ábra.

amikor egyben a Q_1 csoportnak is tagjai” — vagyis az A ügynök számára a hozzáférés most engedélyezve van. A második értelmezés tehát a $Q_1 \cup \tilde{Q}_2$ halmazba tartozó ügynökök számára tesz lehetővé hozzáférést (9.6 b) ábra).

Előfordulhat továbbá, hogy a hozzáférést azokra az ügynökökre akarjuk korlátozni, amelyek egyidejűleg tagjai mindkét csoportnak. Ezek halmaza $Q_1 \cap Q_2$, és pillanatnyilag csak a közös részbe eső ügynökök felsorolásával tudjuk megadni.

A gazdaságos hozzáférési listák kialakítását tehát a következő modell megvalósításával érhetjük el. (Egyelőre feltételezzük, hogy egyetlen hozzáférési mód létezik, és ezt engedélyezzük vagy tiltjuk le.)

- Az ügynökökből csoportok képezhetők, amelyekre azonosítójukkal hivatkozunk. A rendszer minden ügynökét egyelemű csoportnak tekintjük, és mellettük legalább az összes ügynökök csoportját jelölni tudjuk.
- Minden egyes ügynökhöz meghamisíthatatlanul hozzárendeljük az összes olyan csoport azonosítóját, amelynek tagja. (A védelmi rendszer tehát tévedésmentesen meg tudja állapítani, hogy egy ügynök egy adott csoportnak tagja-e.)
- A hozzáférést **előírások** formájában rögzítjük. Az előírás lehet **engedélyező** (felhatalmazás) vagy **tiltó** (kizárás) — ez az előírásban egyetlen csoportra vonatkozik. A Q csoportra érvényes tiltó előírás azonos megszorítást jelent, mint a Q kiegészítő csoportra érvényes engedélyező aláírás.
- A hozzáférési lista az adatelemek használatát engedélyező és tiltó előírások metszeteinek és uniójának kombinációjaként írja elő. Figyelembe véve a tiltó és engedélyező aláírások fenti megfeleltetését, a hozzáférés szabályozása megadható a rendszerben nyilvántartott ügynökcsoporthoz és a kiegészítő halmazukból képzett kifejezéssel.

Ahhoz, hogy ez a modell „működőképessé” váljék, rögzítenünk kell a felhasznált halmazelméleti műveleteknek és a segítségükkel képezhető halmazképző kifejezéseknek a tényleges ábrázolási módját. A továbbiakban — első lépésként — lehetőségeinket a diszjunktív normál formára korlátozzuk.

Egy tetszőleges ügynökhalmoz tehát

$$W = \bigcup_{i \in I_W} \left(\bigcap_{j \in J_W} Q_{ij} \cap \bigcap_{k \in K_W} \bar{Q}_{ik} \right) \quad (9.1)$$

alakban állítható elő.

Az első észrevétel, amelyet ehhez a választáshoz hozzá kell fűznünk, hogy az általánoság nem szenved csorbát: minden halmaz, amely az $S = \{Q_i\}$ és az $\bar{S} = \{Q_i \mid Q_i \in S\}$ elemeinek uniója és metszete segítségével előállítható, leírható ebben a formában is. Ismert tény továbbá, hogy ha két halmaz, W és W' a rendszer előre megadott ügynökcsoportjaiból a diszjunktív normál alak segítségével előállítható, akkor uniójuk $(W \cup W')$, metszetük $(W \cap W')$, illetve kiegészítő halmazaik ($\bar{W}$ és $\bar{W}'$) ugyanebben az alakban előállíthatók. A választott előállítási mód hátránya az, hogy a legtömörebb megadási módszerrel elérhetőnél esetenként terjedelmesebb ábrázoláshoz vezethet, mert az ismétlődő elemek összevonásához szükséges zárójel struktúrát nem alkalmazhatjuk. A diszjunktív normál forma azonban elegendően hajlékony ahhoz, hogy az esetek többségében tömör leírásokat kapjunk, és megvalósítása is igen gazdaságos.

A (9.1) kifejezésben a metszet- és az unióképzés explicit jelölését elhagyhatjuk: a csoportokat, amelyek egy zárójelpáron belül követik egymást, a metszetképzés kapcsolja össze, a zárójelpárok közötti művelet pedig az unióképzés. Mivel a diszjunktív normál formában a zárójelek rekurzív használata nem megengedett (minden nyitó zárójel egy végzárójel követ, mielőtt egy újabb nyitó zárójel megjelenne), a zárójel struktúrát előírásenként egyetlen, az ún. **zárójelbit** segítségével ábrázolhatjuk. Ha ennek értéke 0, akkor egy metszeten belül vagyunk, ha viszont 1, akkor a szóban forgó elemet végzárójel előzi meg. Ez a sorozat képezi a hozzáférési listát, egy példát rá a 9.7 ábrán láthatunk. Ugyancsak egy, ún. **előírásbiten** ábrázoljuk az engedélyezést (1) vagy tiltást (0) is.

A hozzáférés ellenőrzése annak megállapításából áll, hogy az adott A ügynök eleme-e a hozzáférési listaként megadott kifejezéssel előállítható ügynökhalmoznak.

Álprogram formájában:

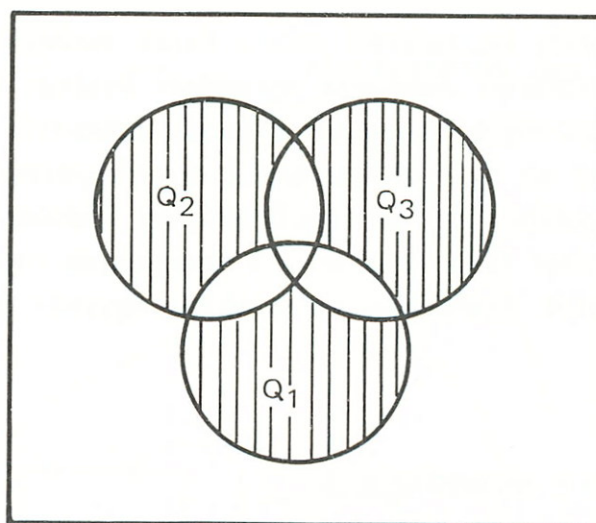
HOZZÁFÉRÉS–ELLENŐRZÉS:

```

1  engedély := YES;
2  while listaelem nem üres do begin
3      if zárójelbit = 1 then
4          if engedély = 1 then
5              exit;
6          else
7              engedély := YES;
8          endif;
9      endif;
10     if engedély = YES then
11         if előírás engedélyező then
12             if  $A \notin Q$  then engedély := NO endif;
13         elseif előírás tiltó then
14             if  $A \in Q$  then engedély := NO endif;
15         endif;
16     endif;
17 end;
18 exit;
```

A hozzáférés-ellenőrzés végeredményét az engedély Boole-változójának kilépés utáni értéke képviseli, amelyet az ellenőrzés kezdetén beállítunk (1), azzal a feltételezéssel,

$$\begin{aligned}
 & Q_1 \cap \tilde{Q}_2 \cap \tilde{Q}_3 \cup \\
 & \cup \tilde{Q}_1 \cap Q_2 \cap \tilde{Q}_3 \cup \\
 & \cup \tilde{Q}_1 \cap \tilde{Q}_2 \cap Q_3
 \end{aligned}$$



Q ₁	1	0
Q ₂	0	0
Q ₃	0	0
Q ₁	0	1
Q ₂	1	0
Q ₃	0	0
Q ₁	0	1
Q ₂	0	0
Q ₃	1	0

Engedélyezés / tiltás

Zárójelbit

9.7 ábra.

hogy a hozzáférési listán legalább egy elem található. A listán elemenként haladunk végig (2). Mikor a metszetsorozat végéhez érünk (3) és a hozzáférés engedélyezve van (4), az ellenőrzést befejezzük (5), mert az engedélyezéshez elegendő, ha az ügynök az unió legalább egy tagjának eleme. Ellenkező esetben (6) az engedély változót ismét beállítjuk (7). Egy metszetsorozat soron következő elemét csak akkor vizsgáljuk meg, ha az ügynök a metszetsorozat minden egyes korábbi elemének tagja volt (10). (Ha a metszet a sorozat egy pontján üressé válik, további elemek nem változtatnak rajta, és így vizsgálatuk elhagyható.) A vizsgálat engedély értékét NO-ra állítja, ha a vizsgált csoportnak az adott engedélyező előírás esetén nem tagja (11, 12) vagy ha tiltó előírás esetén tagja (13, 14).

Különböző hozzáférési módok (olvasás, írás, végrehajtás) is egyszerűen beilleszthetők ebbe az ábrázolási formába. Ehhez mindössze az szükséges, hogy az egyes listaelemekben minden hozzáférési módhoz külön engedélyező bit tartozzon, és a HOZZÁFÉRÉS–ELLENŐRZÉS végrehajtásakor a 11 és 13 sorban az ügynök által gyakorolni kívánt hozzáférési mód engedélyezésbitjét vizsgáljuk.

A többféle hozzáférési módú listák összeállítását elvben úgy képzelhetjük el, hogy az egyes hozzáférési módokra megadott listákat összevonjuk. Ehhez először az egyedi listákat úgy bővítjük, hogy az engedélyezésbittől eltekintve azonosak legyenek. Egy lista bővítésekor az új metszetelemek engedélyezésbitjét 1-re állítjuk, az új unióelemeknek pedig legalább egy (az egyöntetűség kedvéért azonban minden) metszetelemében az engedélyezésbitet 0-vá tesszük. Az összevonás most már abból áll, hogy az azonos listaelem-sorozatok egyikében elemenként egymás mellé gyűjtjük az egyedi engedélyezésbiteket.

9.4 Védelmi alrendszerek

Van a teljes algoritmikus védelemnek egy olyan eleme, amellyel említésen kívül ez ideig nem foglalkoztunk. Ezt az elemet azok a hozzáférési alapműveletek alkotják, amelyeket az adatelemek elérésekor, illetve a hozzáférési jogok megadásakor használunk. A példaként felhozott hozzáférési műveletek ezenkívül tartalmilag egyszerűek is voltak: adatelemek létrehozása, megsemmisítése, írása, olvasása, végrehajtása programként. Egyszerűségüknél fogva ezek nem irányítják figyelmünket arra a nehézségre, amely bonyolultabb hozzáférési műveletek esetén hamar nyilvánvalóvá válik. Erre egyszerűbb példa a következő.

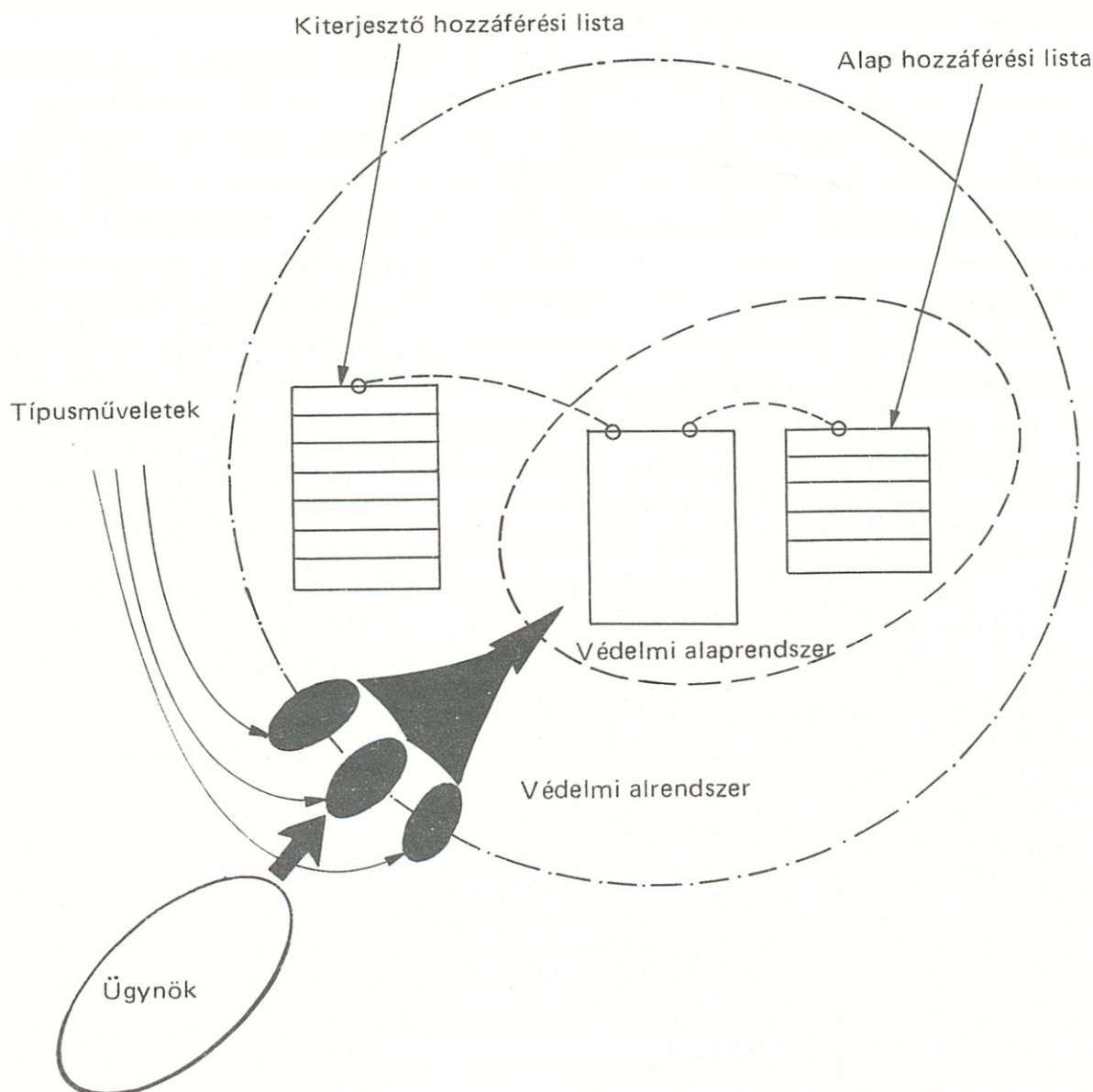
Egy számítógépes rendszerben a különböző adatelemeket tartalmuk, felhasználásuk stb. szerint csoportosítva, könyvtárakban tároljuk és a könyvtárak tartalmát katalógusokban rögzítjük. Egy-egy katalógus azonos szerkezetű tételekből áll, amelyek adatelem-nyilvánartáson kívül egyéb információk, illetve adatok közvetlen tárolására is felhasználhatók. Egy katalógus védelmének szervezője azonban megoldhatatlan dilemma elé kerül: ha a katalógust egyetlen adatelemnek tekinti, akkor egy felhasználó jogai a teljes katalógusra és nem csak az őt illető katalógustételekre vonatkoznak, ha viszont a katalógustételeket egyenként látja el védelemmel, a védelem költsége (tárkapacitás-igénye) nő meg elfogadhatatlan mértékben.

Egy másik, az eddigi modellünkkel csak gazdaságtalanul megvalósítható feladattal találkozhatunk adatbázisok védelmének megszervezésekor, ha a védelmi feltételek az egyes rekordok tartalmától függenek. (Például egy vállalat személyzeti adatbázisa esetén, ha a hozzáférési szabály függ a havi jövedelemtől. Meglehetősen gyakori ugyanis, ha egy bizonyos jövedelemszintet meghaladó alkalmazottak adatait tartalmazó rekordokhoz csak egy szűkebb felhasználói kör férhet hozzá.)

Ezeket és az ezekhez hasonló védelmi feladatokat csak megbízható ügynökök segítségével oldhatjuk meg hatékonyan. A katalógus példában egyetlen speciális ügynök kivételével a katalógust a felhasználók elől elzárjuk. A katalógus csak az előbbi speciális ügynök szolgáltatásainak igénybevételével használható, a hozzáférés feltételeit ez az ügynök ellenőrzi. Az ügynök a katalógus írása és olvasása helyett specializált műveleteket valósít meg: tételkeresés/olvasás, tételbeírás, tételtörlés stb. Ez a speciális ügynök nem csak egyetlen meghatározott katalógushoz tartozik, az összes azonos szerkezetű katalógus kiszolgálására felhasználható.

A védelmi modellünk tehát a következő lehet (9.8 ábra).

Az adatelemek belső szerkezetük és megengedett hozzáférési műveleteik szerint **típusokba** sorolhatók. Egyetlen rendszeren belül is számos adatelemnek lehet ugyanaz a típusa. Minden típushoz a rá jellemző **típusműveletek** tartoznak, egy adatelem csak a



9.8 ábra.

típusának megfelelő típusműveletek felhasználásával érhető el. Minden típusnak van egy **ábrázolása** is, amely meghatározza, hogy a szóban forgó típusú adatelemet hogyan lehet más típusú adatelemből felépíteni. A katalógus például egyetlen szegmensből áll, egy könyvtár (mint típus) azonban egy katalógus-adatelemet és pl.: számos szövegállományt tartalmaz. A típusműveleteket megvalósító ügynök a típusábrázolásában részt vevő adatelemekhez a típusműveleteiken keresztül hozzáférhet, míg ugyanez a többi ügynökök számára közvetlenül tilos.

Mint látjuk, a rendszer hierarchikus, a hierarchia alján a rendszerben definiált alaptípusok találhatók, a további típusok ezekre és egymásra épülnek (rekurzív hivatkozási lánc azonban nem szerepelhet!).

9.5 Az engedély-visszavonás módszerei

Korábban már jeleztük, hogy az engedély alapú rendszerekben a hozzáférés jogosságának ellenőrzése jelentősen gyorsabb, mint a felhatalmazást (hozzáférési listát) alkalmazókban. És mi több, azt is láttuk, hogy egyes alkalmazási helyzetekben emiatt csak az engedélyek használata elfogadható (lásd 9.1 Virtuális tár).

Az engedélyekre három műveletben hivatkozunk:

Hozzáférés (címzés) — Az adatelemekre vonatkozó minden hivatkozás engedélyeken keresztül történik. Az ügynökök a felhasznált adatelemnek még az azonosítóját sem ismerik, az magának az engedélynek a része. A hozzáférésben tehát az engedély nemcsak a hozzáférési mód jogosságának ellenőrzésére, hanem címzésére is szolgál. Mivel az engedélyeket leggyakrabban ebben a műveletben használjuk, megvalósítása hatékony kell, hogy legyen. (Ezzel a kérdéssel a 9.1 és 9.2 fejezetben már foglalkoztunk.)

Továbbadás (másolás, szétosztás) — Az engedélyek használatának lényeges eleme, hogy az ügynökök egymásnak engedélyeket továbbadhatnak. Engedély alapú rendszerekben ez az egyetlen mód közös használatú adatelemek kijelölésére. Engedély-továbbadás-kor egyrészt fel kell készülni arra, hogy küldője később visszavonhassa, másrészt meg kell akadályozni, hogy egy engedély terjedése során újabb hozzáférési jogokat szerezhessen. Ez utóbbi a **hozzáférési jog korlátozásának elve**: egy engedély másolataiban a hozzáférési jogok köre nem bővíthet (csak szűkülhet vagy változatlan maradhat).

Visszavonás — Egy korábban továbbadott engedély kölcsönzését a továbbadója megszünteti.

Az engedélyvek használatával kapcsolatban tehát két olyan tisztázásra váró kérdés is felmerül, amelyeket a korábbiakban még nem tárgyaltuk meg:

- Hogyan különböztetjük meg az engedélyeket az egyes ügynökök adatmezejében tárolt egyéb adatoktól; és
- hogyan tesszük az engedélyeket visszavonhatókká?

Az engedély-visszavonás természetes követelmény: egy számítógépes rendszer élete során az adatelemek közös használatára kötött egyezségeket a kölcsönvevő fél megszegheti, a kölcsönző bizalma partnerével szemben megváltozhat. Engedély-visszavonás szükséges akkor is, ha az adatelemek megosztása csak átmeneti jellegű, vagy ha az engedély kiadásában elkövetett tévedés javítására lehetőséget akarunk adni (ami egyben természetes követelmény is).

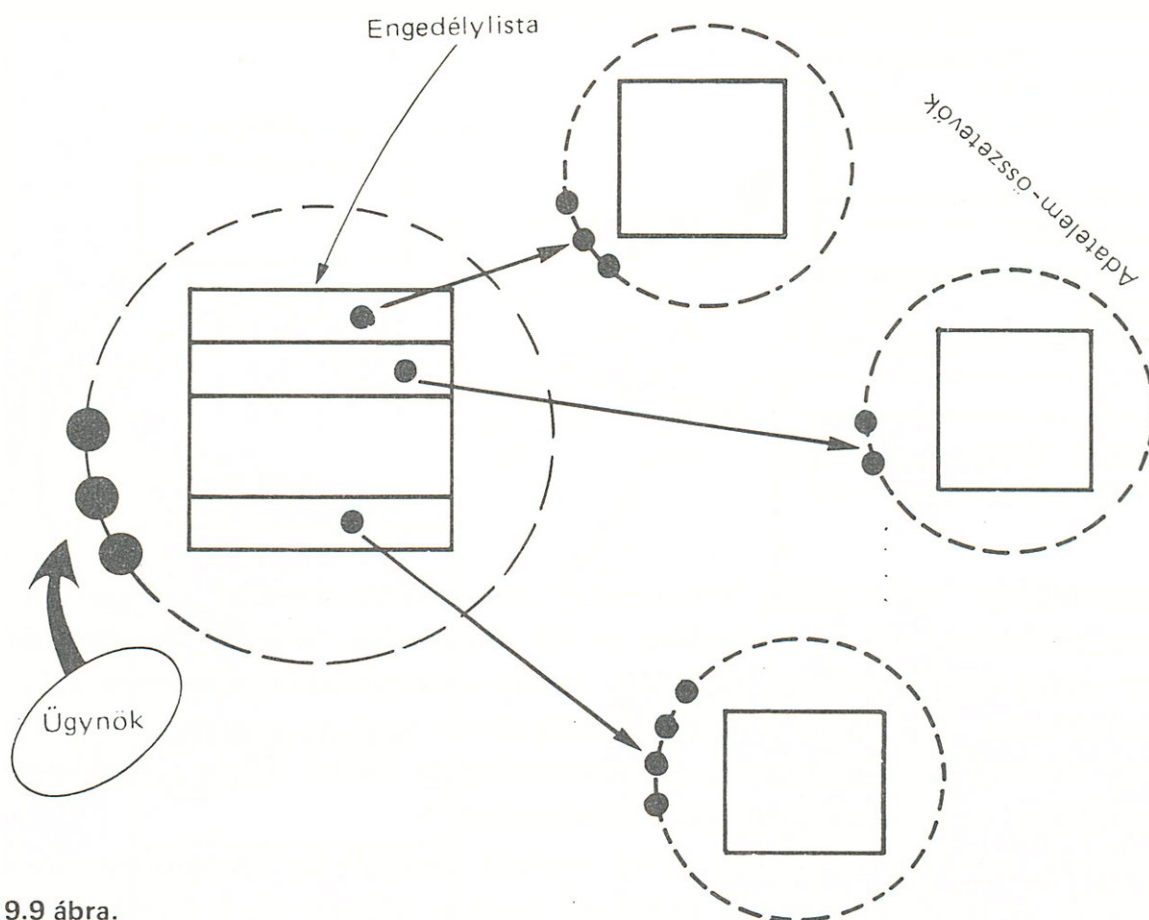
9.5.1 Engedélyek tárolása

Az engedélyvek védelme két feladat megoldását igényli. Meg kell akadályozni, hogy

- az ügynökök az engedélyeket ellenőrizhetetlenül módosítsák, valamint hogy
- a közönséges adatok engedélyként értelmezhetők legyenek.

Az engedélyeket tartalmazó tárolóelemeket tehát el kell különíteni a „közönséges” adatokat tartalmazóktól.

A jelenkori számítógépek zömében, amelyek strukturálatlan operatív tárral, illetve „típus nélküli” tárolóelemekkel rendelkeznek csupán, ez csak egyetlen módon lehetséges. Az engedélyeket ügynökönként egy-egy speciális szegmensben (az engedélylistán) tároljuk, amelyhez maga az ügynök nem férhet hozzá. Az engedélyvek használatára vagy módosítására vonatkozó minden művelet a működtető rendszer szolgáltatása. Használatuk előtt az engedélyvek előbb a szegmensből speciális regiszterekbe (pl.: szegmens-bázis-regiszterek) kerülnek. A berendezés (központi egység) szintjén ez biztosítja hatékony elszigetelésüket az egyéb adatoktól. A védelmi lánc eleme az is, hogy az engedélyeket mozgató (szegmens-bázisregisztereket töltő) utasítások kizárólag a központi egység speciális, csak a működtető rendszer számára hozzáférhető állapotában alkalmazhatók. A védelmi stratégia előírható hozzáférési listákkal vagy engedélyekkel — amelyeken



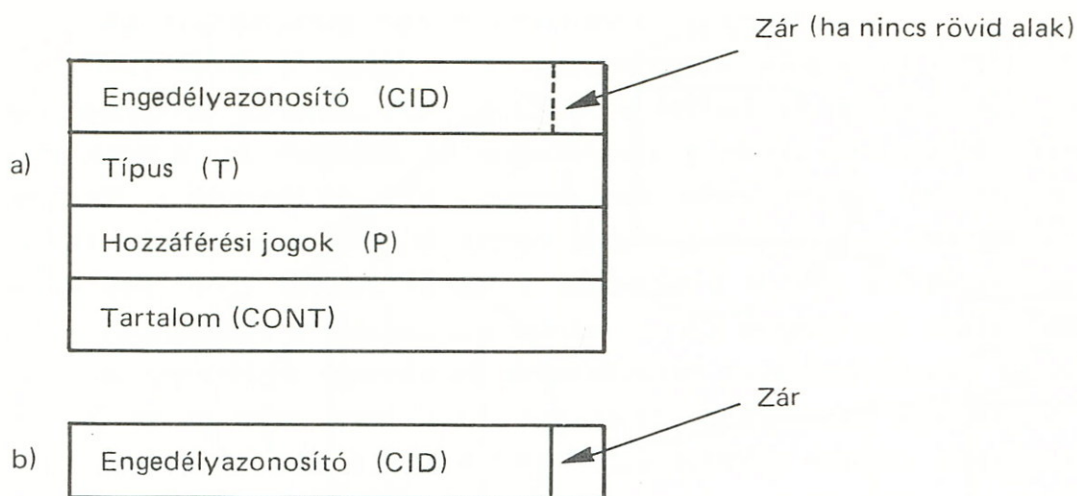
9.9 ábra.

a hozzáférési jogok a típusműveletek segítségével vannak kifejezve —, illetve a típusműveletekbe beépített tartalomfüggő ellenőrzések formájában. A típusábrázolásban célszerű azonban az engedélyekre támaszkodni (9.9 ábra). Egy összetett ábrázolású adatelem összetevőit ez esetben egyetlen olyan adatelemmel fogjuk össze, amely az összetevő elemekre vonatkozó engedélyeket tartalmazza. Ezt a leíró adatelemet védjük azután a típus szerinti védelem módszereivel.

A típusképzéssel, típusműveletekbe épített ellenőrzésekkel a védelmi rendszert fokozatosan és célszerűen fejleszthetjük. Bármely felhasználónak joga van például új típust és ezzel együtt új **védelmi alrendszert** meghatározni. Ezzel egyidejűleg meg kell adnia a típusműveleteket és az adott típusú adatelemek előállításának, törlésének (ezek lényegében szintén típusműveletek) eszközeit.

Jelölhető (tárolóelemeket tartalmazó) operatív tárú (tagged memory) számítógépekben a tárolóelemeket a tartalmuknak megfelelő típus jelével megjelölhetjük. (Ez általában tárelemenként 2–4 bit terjedelmű, és 4–16 típus különböztethető meg vele.) A típusjel alapján a központi egység az utasításértelmezés és -végrehajtás során ellenőrzéseket, típus szerinti műveletkiválasztást stb. végez. Jelölhető operatív tárban mód nyílik az engedélyek megkülönböztetésére, ezért az ügynökök adatmezejében az engedélyek az adatok között megkötés nélkül tárolhatók. A továbbiakban jelölhető operatív tárat és egyetlen JEL bitet feltételezünk. A JEL = 1 engedélyt, a JEL = 0 egyéb adatot jelöl. (Ez természetesen csak a tárgyalást egyszerűsíti. Minden eredmény és algoritmus a külön szegmensben tárolt engedélyekre is érvényes.)

Az engedélyek belső szerkezetét a 9.10 a) ábra mutatja. Minden engedélynek egy egyértelmű **engedélyazonosítója** van (CID), és valamilyen objektumra vonatkozik, amelyre a **tartalom** (CONT) mező hivatkozik. Az engedély a megnevezett objektum **típusát**



9.10 ábra.

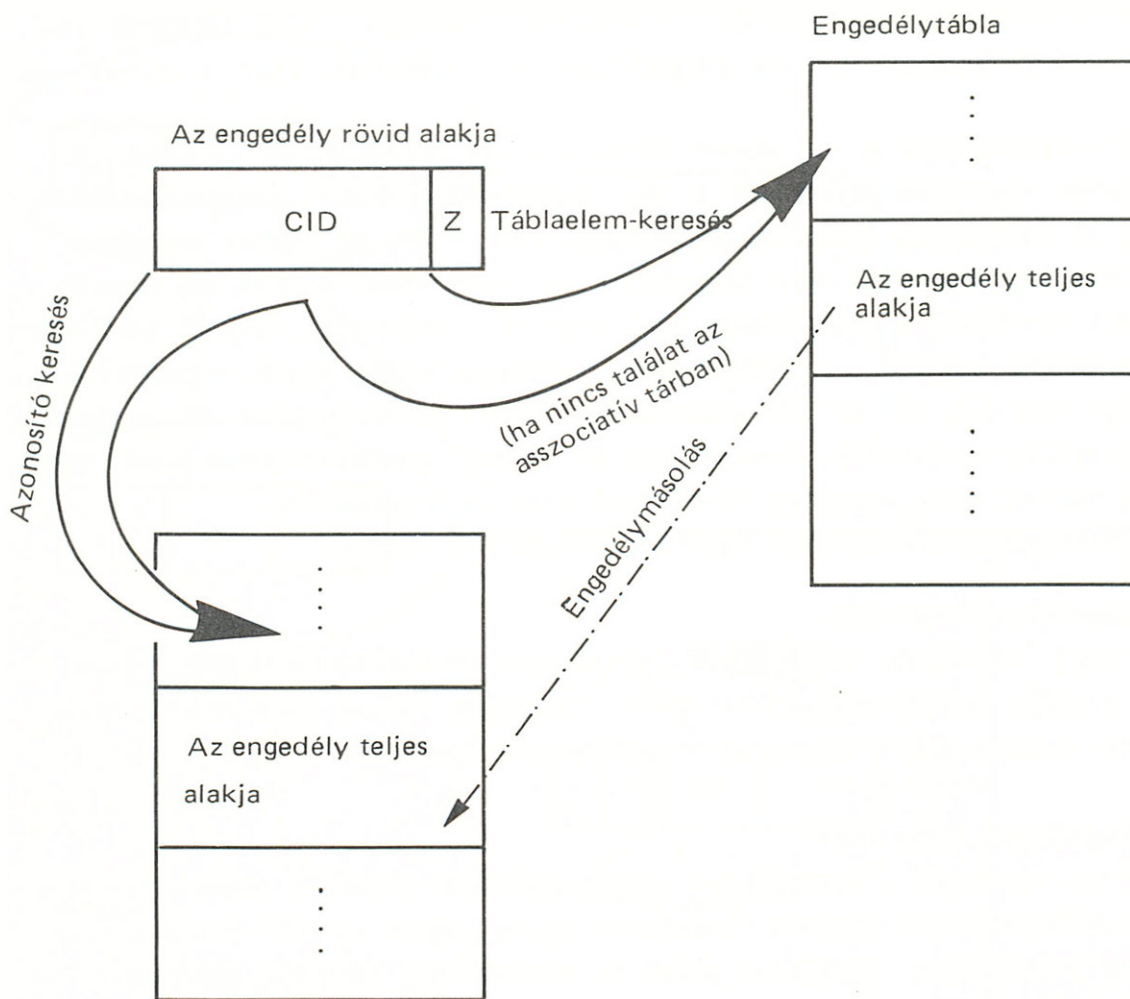
(T) és a típusműveletekben (9.4 fejezet) kifejezetten **hozzáférési jogokat** (*P*) szintén tartalmazza. Az engedélyeknek ez az ábrázolása az ún. **teljes alak**, emellett minden engedélynek van egy **rövid alakja** is (9.10. b) ábra), amely csak az azonosítóját és az ún. **ZÁR** (*Z*) bitet tartalmazza. Ez utóbbi 1 értéke lehetetlenné teszi az engedély bármilyen értelmű módosítását (kivéve természetesen a megsemmisítését), illetve egyes – az engedélyek felhasználását érintő – műveletekben való alkalmazását.

Az engedélyeknek csak ezt az utóbbi, rövid alakját tároljuk és mozgathatjuk szabadon. A teljes alakot az **engedélytáblában** (listában) tartjuk nyilván, és a rövid alakból vett azonosítójával indított asszociatív kereséssel férhetünk hozzá. Ezt a táblaelemkeresést leggyakrabban terítő-léptető függvénykombinációval (hashing) valósítjuk meg. Engedélyek sűrűn való alkalmazásakor ezt a leképezési folyamatot asszociatív tárral gyorsíthatjuk meg éppen úgy, mint a virtuális tár (9.1 fejezet) esetében tettük (9.11 ábra).

Az engedélyek elsősorban adatelemekre hivatkoznak, **CONT** mezejükben egy adatelem-azonosító van. Ez az azonosító azután közvetlenül (pl.: címként felhasználva) vagy közvetetten (pl.: az adatelemek leírását tartalmazó táblán keresztül) választja ki a hivatkozott adatelemet. Ha az adatelem típusa a rendszer alaptípusainak egyike, akkor tartalma adat, ha összetett típus (tehát védelmi alrendszerrel állunk szemben), akkor pedig a típus ábrázolását (9.4 fejezet) meghatározó engedélyeket tartalmazza.

Az adatelemeken kívül az engedélyek hivatkozhatnak egymásra is. Ilyenkor **CONT** mezejükben egy másik engedély azonosítója található. Így engedélyek között kapcsolat hozható létre, amely általában függőségi (egy engedély sorsa, hatása valamilyen módon függ az általa megnevezett másik engedélytől). Az engedélyek ilyen értelmű használatának természetes következménye, hogy az adatelemek és engedélyek azonosítóit ugyanabból a halmazból választjuk.

Végül meg kell jegyeznünk, hogy bár a továbbiakban a 9.10–9.11 ábrán bemutatott engedélyábrázolást vesszük alapul, eredményeink átvihetők az egyéb (egyszerűsített vagy specializált) engedélytárolási formákra is, és legtöbbször a változás csak egyes műveletek hardverből szoftverbe való mozgatásában nyilvánul meg. Az egységes hivatkozás kedvéért a 9.1 táblázatban feltüntetettük a további algoritmusok megfogalmazásában felhasznált jelöléseket (elemi műveleteket) és azok jelentését (funkcióját).



9.11 ábra.

9.1 táblázat.

CID	engedélyazonosító
T	elemtípus
P	hozzáférési jogok
CONT	elemtartalom
elem(I)	az I azonosítójú engedély-táblabeli elem
CID-keres (I)	I azonosítójú elem keresése az asszociatív tárban. Ha ilyen létezik, akkor a keresés azon áll meg, ha nem, akkor az első szabad asszociatív-tár-elemen
CONT-keres (I)	ugyanaz, mint a CID-keres, csak a CONT mezőt használja

9.5.2 Az engedély-visszavonás változatai

Az engedély-visszavonással szemben támasztható követelmények szemléltetésére a 9.12 ábrán egy rendszer különböző hozzáférési tartományait és egyetlen engedély (CO) másolatai létesítette kapcsolatukat tüntettük fel. Az eredeti engedély az A tartomány ré-

sze, másolatai innen jutottak el terjesztéssel a többi tartományba. (A *C* tartományba különböző utakon két példányban is.) Az engedélyek terjesztésének útját a nyilak mutatják.

Az engedélyek visszavonásának az ábrán láthatóan van egy „durva” módja, az adatelem azonosítójának megváltoztatása (és a régi adatelemtábla-tétel megsemmisítése). Ez azonban csak az *A* tartomány számára engedhető meg, mert az összes engedély egyidejű visszavonásával egyenértékű; egy tartomány azonban csak azokat az engedélyeket vonhatja vissza, amelyek a birtokában lévő engedély másolatai. (Az *E* tartomány – illetve a benne működő ügynök – például nem vonhatja vissza egyik engedélypéldányt sem, *B* pedig csak *C3*, *C4* és *C5* visszavonása felett rendelkezik.) Az esetek zömében a totális engedély-visszavonás nem is cél, az engedélymásolatoknak csak meghatározott részét és esetleg csak átmeneti érvénnyel akarjuk visszavonni.

Az alábbiak az engedély-visszavonás hasznos és gyakori formái.

a) **Szelektív engedély-visszavonás**

Egy tartomány az általa terjesztett engedélyek tetszőleges részhalmazát képes visszavonni, miközben a további példányok érintetlenek maradnak. Az *A* tartomány például az *E* tartománytól vissza akarja vonni az engedélymásolatot, de a *B*, *C* és *D* tartományoktól nem.

b) **Részleges engedély-visszavonás**

Az engedély-visszavonás nemcsak a hozzáférési jogok egészére (tehát a teljes engedélyre), hanem ezek tetszőleges részhalmazára (beleértve egyes jogok egyedi visszavonását is) alkalmazható. Egy adatelemre vonatkozó írási és olvasási jog például egymástól függetlenül és egyenként is visszavonható.

c) **Független engedély-visszavonás**

Ugyanannak az engedélynek két különböző úton érkező példányát egy közös kibocsátójuk egymástól függetlenül képes visszavonni. A 9.12 ábrán például a *B* tartomány a *C3* és *C5* engedélyből függetlenül képes hozzáférési jogokat visszavonni.

d) **Tranzitív engedély-visszavonás**

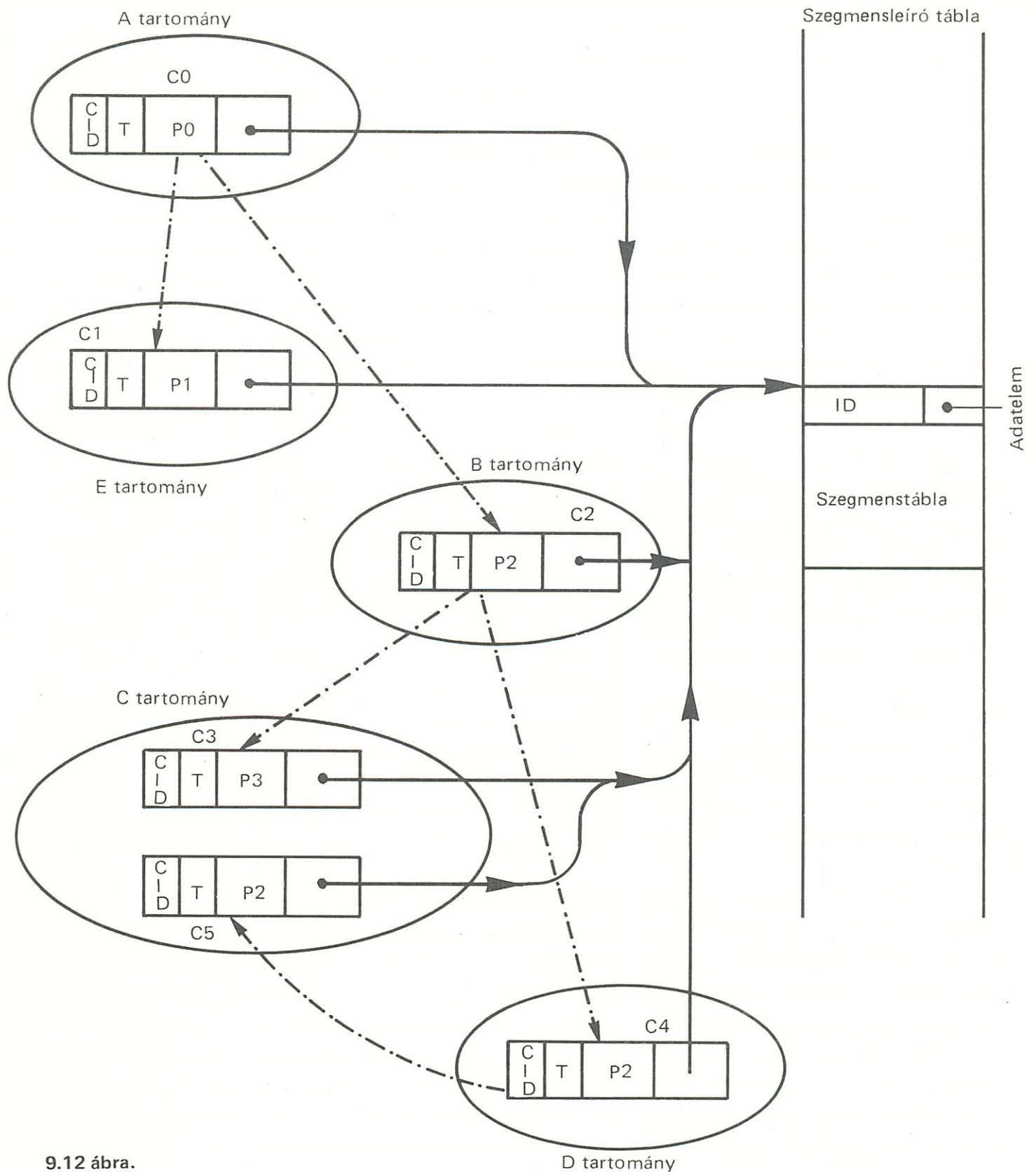
Egy hozzáférési jog (jogok, teljes engedély) visszavonása nemcsak a visszavonó által megnevezett tartományban érvényesül, hanem mindazokban a tartományokban is, amelyek a hozzáférési jogot (jogokat, engedélyt) ez utóbbi tartománytól közvetlenül vagy közvetetten megkapták. Ha például az *A* tartomány a 9.12 ábrán egy hozzáférési jogot visszavon *B*-től, akkor ez eltűnik a *C* és *D* tartományból is (de az *E*-ből nem). A tranzitivitás az engedély-visszavonás szinte kötelező tulajdonsága kell, hogy legyen, hiányában ugyanis több tartomány együttesen kijátszhatná az engedély-visszavonó rendszert. (A kapott hozzáférési jogot a fogadó tartomány átadja cinkostársának, majd az eredeti hozzáférési jog visszavonása után tőle ismét megkaphatja.)

e) **Időleges engedély-visszavonás**

A visszavonás automatikusan érvényteleníthető (tehát ugyanazzal a mechanizmussal, mint amellyel a visszavonás történt). Az időleges visszavonás egyetlen előnye, hogy a tartományoknak nem kell az engedélyterjesztést újra végrehajtaniuk, ha a visszavonás után vissza akarnak adni egyes jogokat. Az időleges engedély-visszavonás hasznossága erősen kérdéses, mert az engedélyterjesztés általában egyszerű és olcsó, a visszavonás (különösen az ideiglenes) pedig viszonylag ritka művelet.

f) **Azonnali engedély-visszavonás**

Egy tartomány a visszavont jogot (engedélyt) már attól a pillanattól kezdve nem hasz-



9.12 ábra.

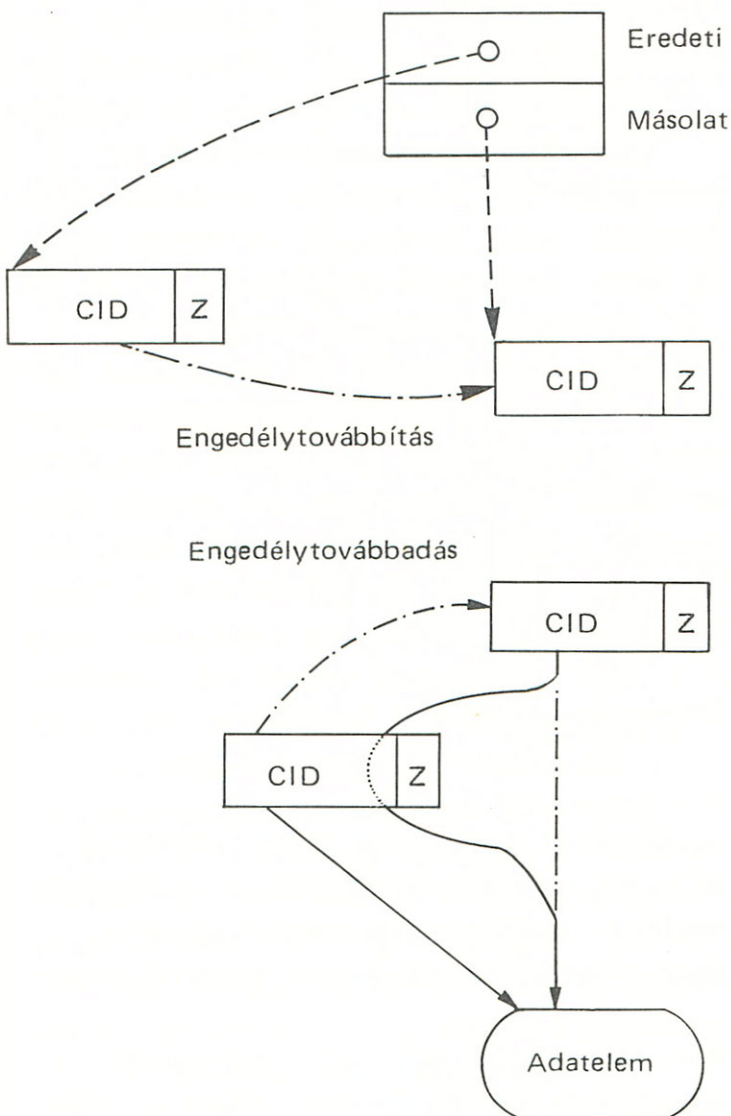
hálhatja, amint a védelmi rendszer a visszavonási igényt tudomásul vette (amikor a rendszer a visszavonási parancsot „megfejtette”). Az engedély-visszavonás és különösképpen az azonnali engedély-visszavonás ellentmondásos adatelem-tartalomhoz vezethet, ezért az adatelemek tartalmi következetességének visszaállításáról külön gondoskodnunk kell.

Az engedély-visszavonás fenti tulajdonságai érthetően különböző „költségűek” a visszavonó rendszer tárhelykapacitás-igénye és a visszavonás kivitelezésének időigénye tulaj-

donságokként változó. A visszavonhatóság azonban mindenképpen bonyolítja és lassítja az engedélyek szétosztását és/vagy használatát. Az engedélyeket kezelő rendszert ezért úgy kell megalkotni, hogy a visszavonhatóság ne legyen kötelező és általános érvényű. Megbízható tartományok, amelyek a birtokukba jutó engedéllyel nem élnek vissza, és használat után megsemmisítik, a visszavonó rendszer közbejötté nélkül kaphatnak tehát engedélymásolatokat. Egy számítógépes rendszerben az ilyen engedély-továbbadások az engedélyforgalom jelentős részét képezik, mivel a működtető rendszer szolgáltatásait megvalósító ügynökök ebből a szempontból megbízhatónak tekinthetők. Egy engedély tulajdonosa természetesen szabadon dönt afelől, hogy visszavonható vagy visszavonhatatlan engedélyt ad-e partnerének. Egy visszavonható engedélymásolat birtokosa azonban a másolat továbbadásakor már kötelezhető a visszavonó rendszer használatára.

Az engedély-visszavonás megvalósításában végleteket képvisel a 9.13 ábrán változatosan szemléltetett két alapelv.

A 9.13. a) ábrán az eredeti és másolati engedély között egy külső adatstruktúra létesít kapcsolatot. Egyetlen engedély és másolata esetén ez egy rekord, amelyből mindkét engedély elérhető. Egyetlen engedély ismételt továbbadásakor az egyes rekordok az ún. **engedélyterjesztési gráf** csomópontjaivá válnak. Az engedélyterjesztési gráffal a következő fejezetben foglalkozunk, már itt kiemeljük azonban, hogy az engedélyekhez képest külső adatstruktúra lévén hozzáféréskor az engedélyek használatát semmiben sem módosítja. Mivel az engedélyeket leggyakrabban adatelem-elérésre használjuk,



9.13 ábra.

az engedélyterjesztési gráfra támaszkodó engedély-visszavonó rendszernek ez előnyös tulajdonsága. Ennek ára azonban — mint látni fogjuk — az engedélyek továbbadásának és visszavonásának megnövekedett időigénye.

A (9.13. b) ábrán az ún. **kapcsolt engedélyeket** mutatjuk be. A visszavonás ezek segítségével azért lehetséges, mert a továbbadott engedélyt közvetlenül függővé tesszük az eredetitől. Bármilyen módosítás az eredeti engedélyben — pl.: egyes jogok letiltása — azonnal érvényesül a másolatban is. Kapcsolt engedélyekkel az engedély-visszavonás egyszerűbbé válik, az engedélyek alkalmazása azonban bonyolódik, és csak gondos tervezéssel, megfelelő berendezéssel (központi egység architektúra) tehető működőképesé. Ez esetben viszont talán a jövő általános megoldását jelentheti. Ezt a módszert is egy önálló fejezetben ismertetjük.

9.5.3 Engedélyterjesztési gráf

A 9.12 ábra példájához tartozó engedélyterjesztési gráfot láthatjuk a 9.14. a) ábrán. Ennek minden egyes csomópontja egy engedélymásolásnak felel meg. A 9.14. b) ábra mutatja az egyes csomópontok szerkezetét. A csomópont EREDETI mezeje a továbbítandó, MÁSZOLAT mezeje a továbbadott engedélyre mutat. Hivatkozásul pl. annak a tárolóelemnek a címe használható, amely a szóban forgó engedélyt tartalmazza. Egy csomóponttól irányított él húzódik egy másik csomóponthoz, ha az előbbi MÁSZOLAT mezejének tartalma megegyezik az utóbbi EREDETI mezejének tartalmával. Ezeket az éleket közvetlen formában az egyes csomópontok ÉLEK mezejében elhelyezett mutatók képviselik, segítségükkel az engedélyterjesztési gráf végigjárása egyszerűsödik. Fontos megjegyeznünk, hogy az engedélyterjesztési gráf minden érdelemleges esetben fa, kivétellel csak akkor találjuk szembe magunkat, ha egy engedélymásolatot ugyanannak az engedélynek egy másik úton érkezett másolatával felülírunk. Ez azonban csak átmeneti helyzetet jelent, mert az átírt engedély csomópontját a periodikus karbantartás során előbb-utóbb töröljük.

Az engedélyterjesztési gráf használatának bemutatásához tegyük fel, hogy a 9.12 ábra példájában a B tartomány néhány hozzáférési jogot vissza akar vonni a D tartománytól. A B tartomány ennek érdekében az engedély-visszavonó rendszerhez fordul, és két paramétert ad át neki: annak az engedélynek a címét, amelynek másolatát kapta meg a D tartomány (loc.C2) és egy bitvektort, amely maszkként alkalmazva határozza meg a visszavonandó hozzáférési jogokat. Az engedély-visszavonó rendszer ezután egy olyan csomópontot keres adatbázisában, amelyben a MÁSZOLAT mező a megadott engedélyre mutat (tartalma loc.C2). Ez a keresés az AB csomópontot találja meg, amely az $AB \rightarrow BD \rightarrow DC$ és $AB \rightarrow BC$ engedélyterjedési utaknak a kezdőpontja. Ezeket azután az engedély-visszavonó rendszer bejárja, és a hozzáférési jogok halmazának kívánt módosítását minden útba ejtett engedélyben elvégzi.

A tranzitív és részleges visszavonást tehát az engedélyterjesztési gráf biztosítja. Segítségével a szelektív és független engedély-visszavonás ugyancsak megvalósítható. Ehhez azonban az egyes tartományoknak a visszavonás körülményeit előre vélelmezniük kell. Ha például az előbbi visszavonásban a B tartomány a $C3$ engedélymásolatot nem akarja visszavonni a C tartománytól, akkor ezt a helyzetet előre kell látnia, és a $C2$ engedély helyett annak egy $C2'$ másolatát kell C -nek átadnia. Mivel a címtartományban $C2$ és $C2'$ különböző címeken található, $C2'$ másolataiban a visszavonás nem történt

Az előzőekben feltételeztük, hogy az engedélyterjesztési gráfot a csomópontokban elhelyezett mutatók segítségével jártuk végig. Engedély-továbbadaskor tehát a keletkező új csomópontot hozzá kell csatolnunk a szóban forgó engedélyterjesztési gráfhoz. Ehhez viszont meg kell keresnünk az éppen másolt engedélyhez tartozó csomópontot, a csomópontok MÁSOLAT mezeinek a végignézésével. Mivel ez időigényes, az engedélymásolás hatékonyságát nagymértékben lecsökkentheti. Átmeneti megoldásként az új csomópont csatolását elhalaszthatjuk, és a rendszer egy háttér folyamatára bízhatjuk. Ez gondoskodik egyben az engedélyterjesztési gráf teljes karbantartásáról is. Az új csomópontok csatolását természetesen a legközelebbi visszavonásig végre kell hajtani. Ha ez nem történt meg, akkor a visszavonás elsőként ezt kezdeményezi. A teljes rendszer természetesen még számos finomságot és hatékonyságnövelő megoldást tartalmazhat. Ezekkel azonban itt már nem foglalkozunk.

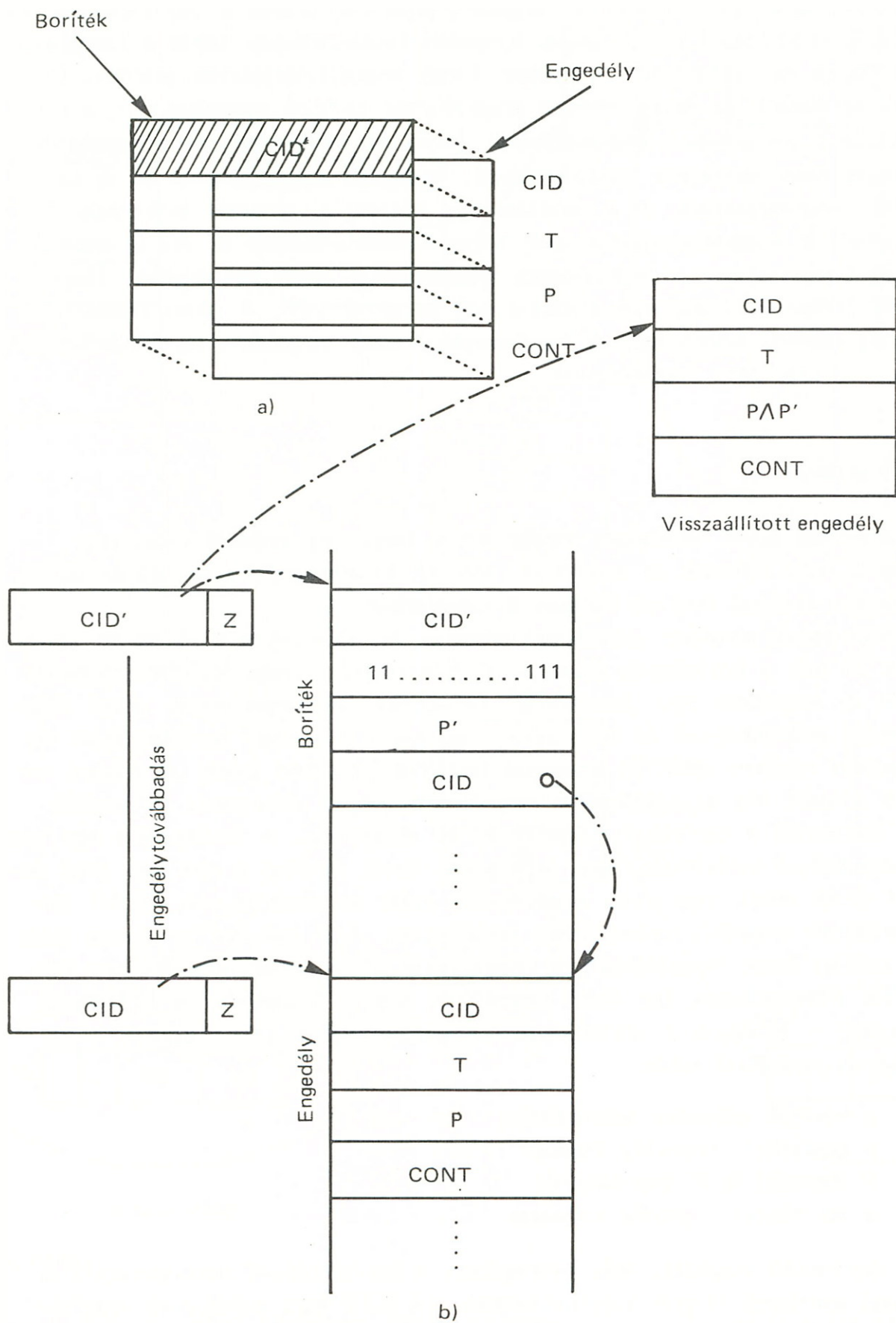
9.5.4 Kapcsolt engedélyek

Az engedély-visszavonás egyik lehetséges módja az is, hogy egy engedélymásolatot olyan szoros kapcsolatba hozunk az eredetivel, aminek következtében ez utóbbi bármilyen megváltoztatása azonnal érezteti hatását a másolatban is.

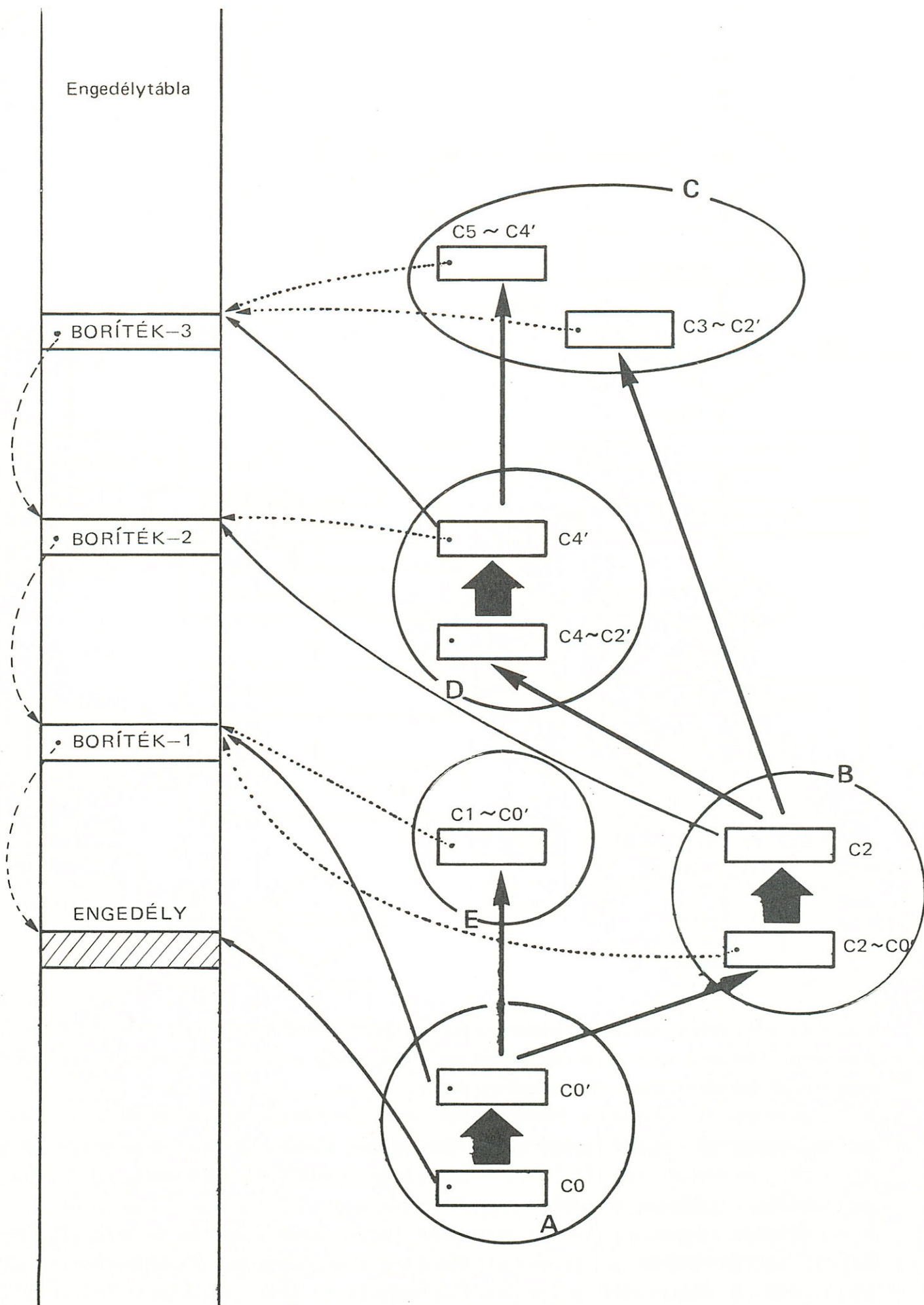
Ezta borítékolásnak nevezett művelettel érhetjük el, amelyet a 9.15 ábrán szemléltetünk. Ennek *a)* fele a borítékolás elvét, a *b)* fele pedig a megvalósítását mutatja. Továbbadás előtt az engedélyt egy „borítékba” helyezzük, amelynek egyes részei átlátszóak, mások pedig átlátszatlanok és információ van rájuk írva. Egy borítékolással létrehozott új engedély részben tehát új adatokat tartalmaz, részben pedig az eredeti engedély megfelelő adatait. Ha az utóbbiakat megváltoztatjuk, a változtatás (a boríték átlátszó részein keresztül) a borítékolt engedélyen is végbemegy. A borítékolás egy speciális formájú engedéllyel valósítható meg, amelynek típusa például a rendszer által felismert $11 \dots 1$ érték lehet. Egy ilyen engedélynek saját azonosítója van, *CONT* mezeje pedig a borítékolt engedély azonosítóját tartalmazza. A *P* mezőbe írt 1-esek átlátszóak (tehát a szóban forgó hozzáférési jog megegyezik a borítékolt engedélyben feltüntetetttel), a 0-k átlátszatlanok (az illető hozzáférési jog – függetlenül a borítékolt engedély tartalmától – hiányzik a származtatott engedélyből). A származtatott engedély visszaállításának szabályai tehát

CID = a boríték engedély azonosítója;
T = a borítékolt engedély típusa;
P = *P* (boríték) & *P* (borítékolt);
CONT = a borítékolt engedély tartalma.

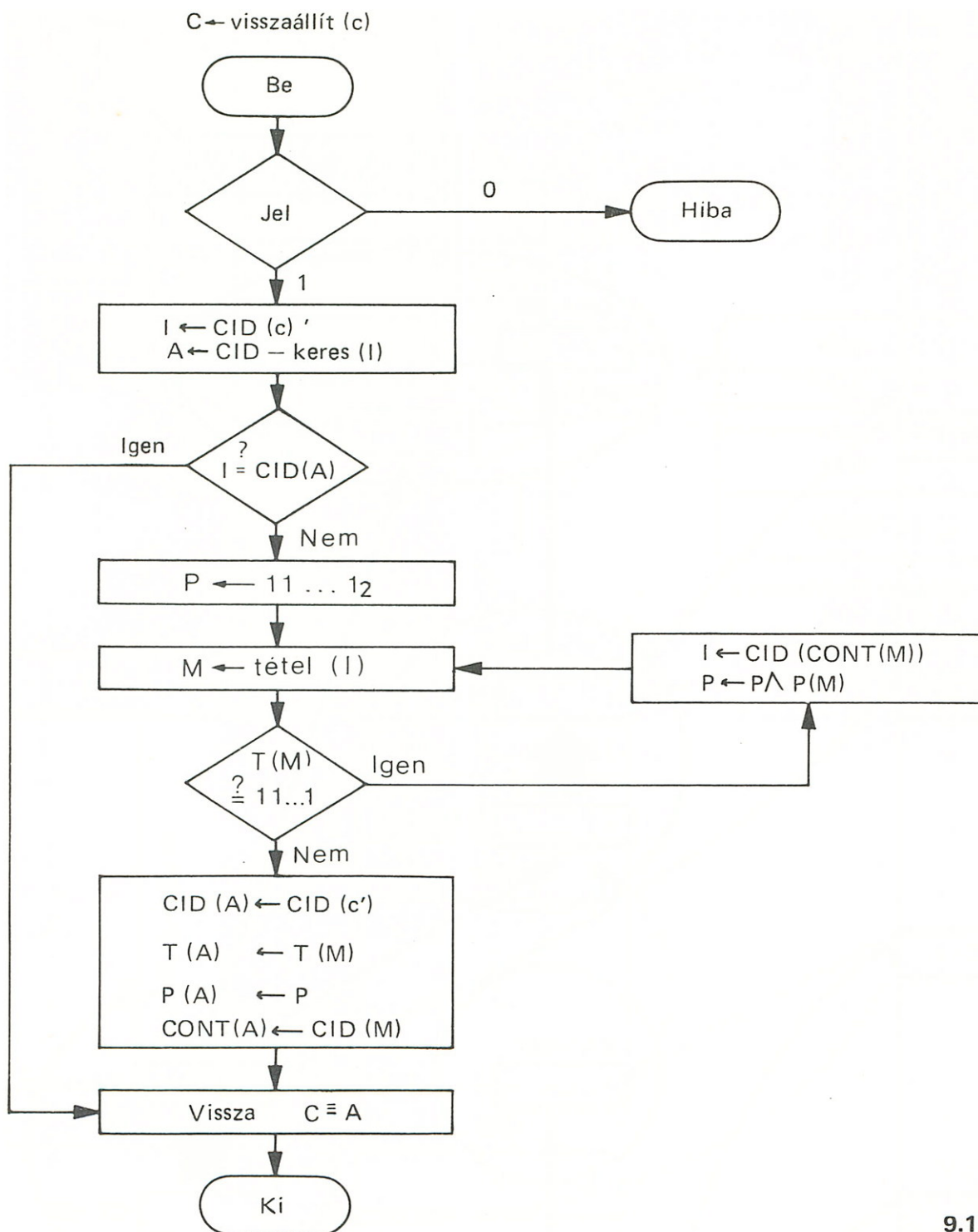
Mivel egy borítékolt engedély már önmagában is borítékolással származtatott lehet, borítékolással sokelemű láncok keletkezhetnek. (A 9.12 ábra példájának megfelelő lánc például a 9.16 ábrán látható.) Ez viszont a származtatott engedélyek alkalmazását jelentősen lelassítja, mert a visszaállítás időigényes műveletté válik. Asszociatív tár alkalmazásával azonban a visszaállítás gyorsítható (lásd 9.11 ábra). A lánc végigkövetése mellőzhető, ha a visszaállított engedély az asszociatív tárban megtalálható. Ha nem, akkor a lánc végigjárása nem kerülhető el, a visszaállítás eredményét azonban beírjuk az asszociatív tárba. A visszaállítás folyamatábráját a 9.17 ábra mutatja.



9.15 ábra.



9.16 ábra.

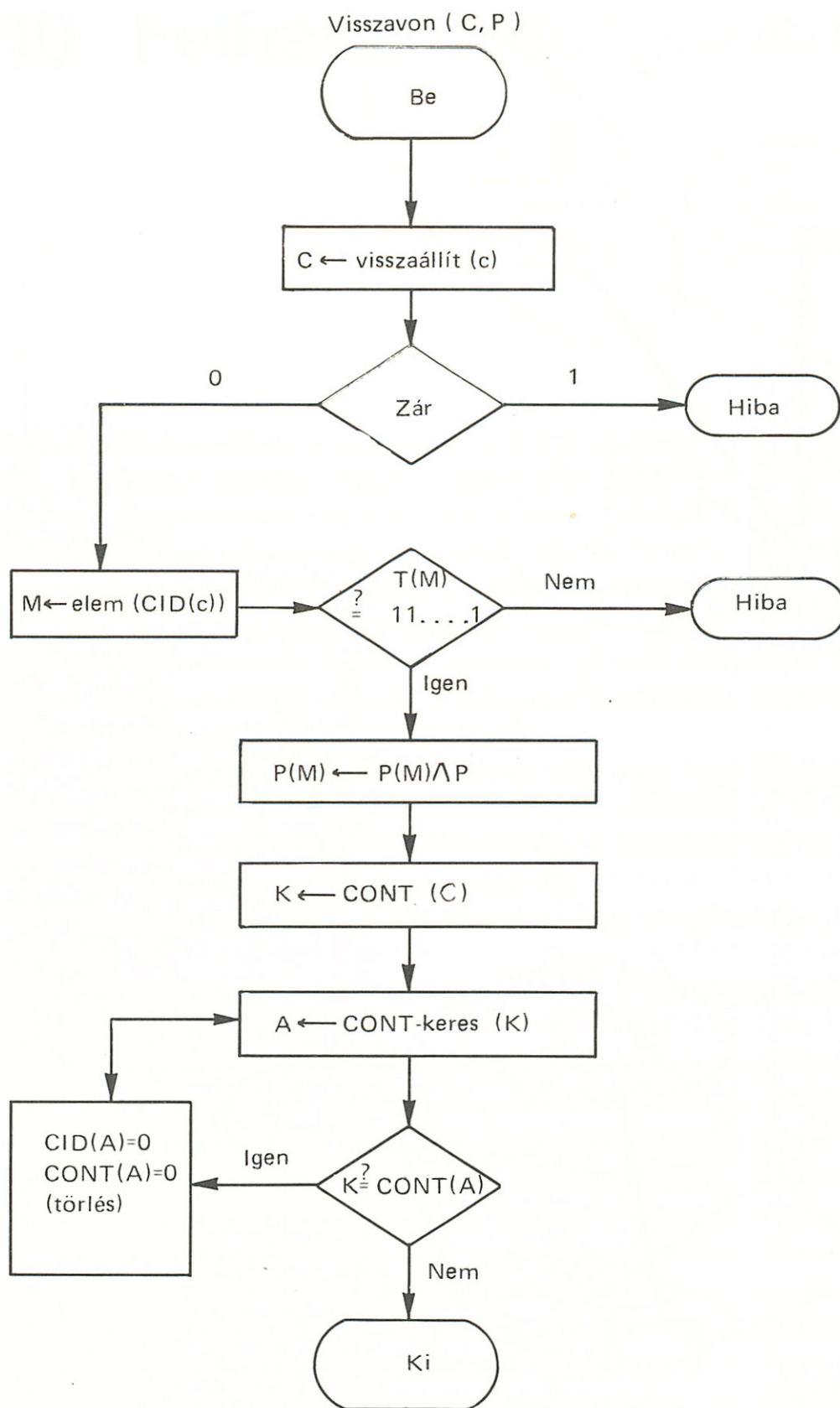


9.17 ábra.

Kapcsolt engedélyekkel az engedély-visszavonás menete ezek után a 9.18 ábrán végigkövethető. Néhány szót érdemes azonban még ejteni a borítékolásnak a visszavonható engedélyek terjesztésében való alkalmazásáról.

Az engedélyek továbbadása mindig rövid alakjuk másolásával történik (9.19 ábra). Mivel az így keletkező engedélypéldányok teljes alakja közös, visszavonási helyzetük azonos: bármelyik példányon keresztül is vonunk vissza hozzáférési joga(ka)t, ez a visszavonás egyidejűleg mindegyik példányra vonatkozik.

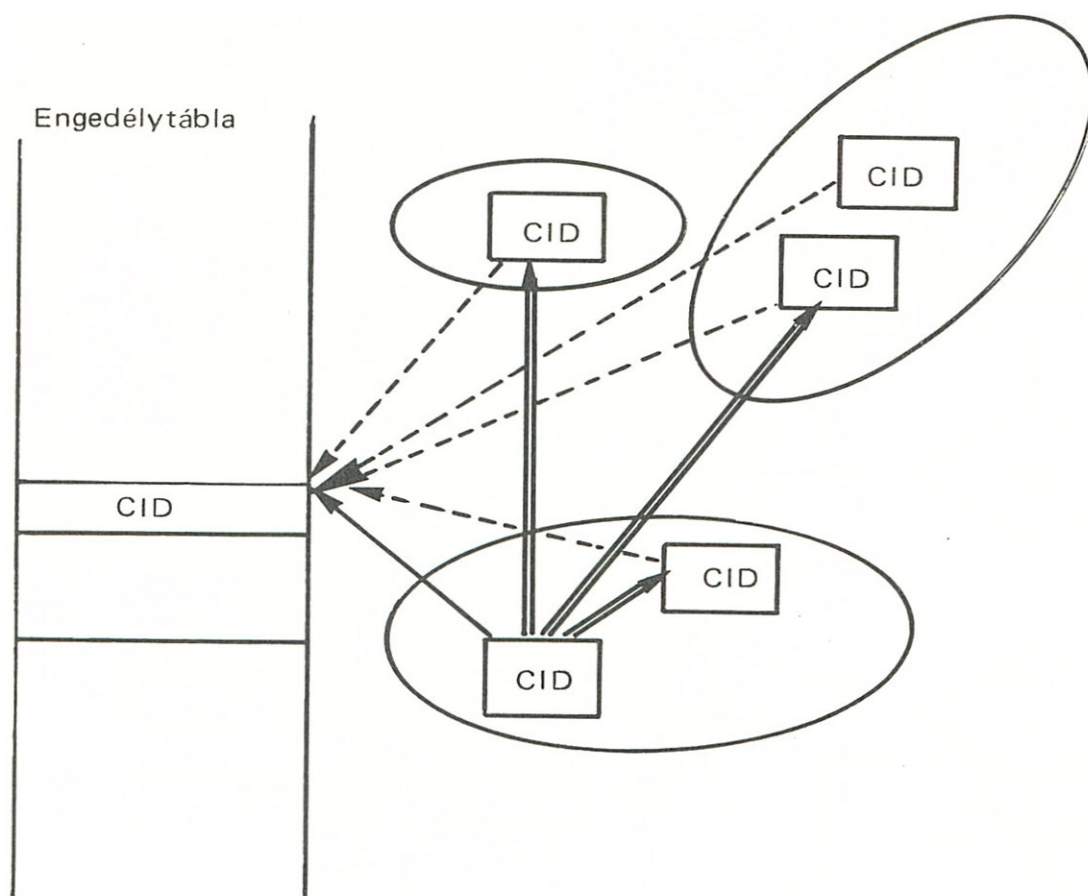
A borítékolás viszont új (bár az eredetitől függő) engedélypéldányt állít elő. Egy engedély két borítékolásával a visszavonás szempontjából független engedélyeket kapunk (9.20 ábra), mert a visszavonást a boríték P mezejébe írt 0-kal valósítjuk meg. Egy engedély visszavonható másolatát úgy kapjuk, hogy az engedélyt borítékoljuk, majd az így



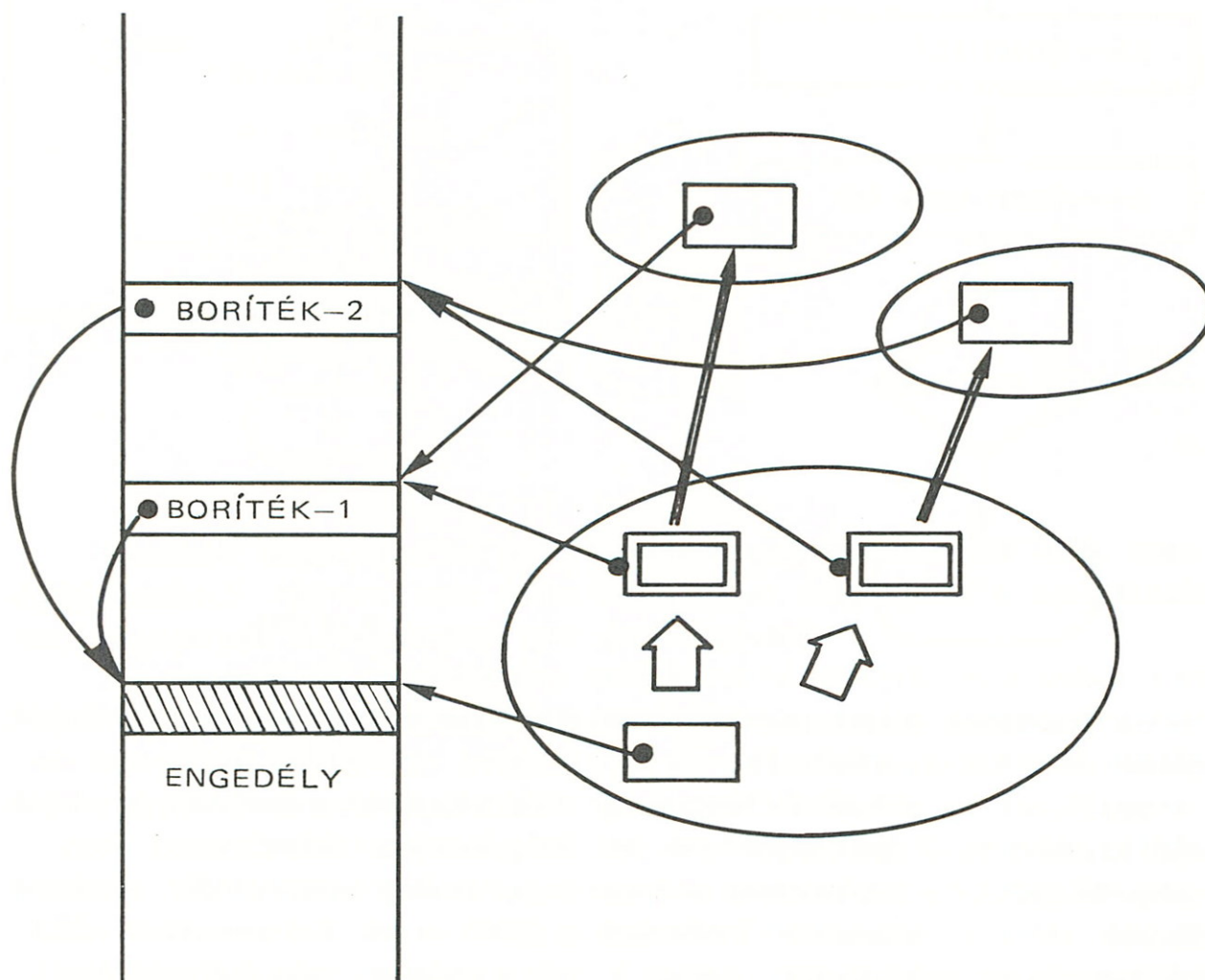
9.18 ábra.

kapott engedély rövid alakjának másolatait adjuk tovább. (Az első példányt megtartjuk, mert a visszavonásban erre kell hivatkozni.)

Az eredeti engedélynek az előbbtől függetlenül visszavonható másolatait egy újabb borítékolással, majd továbbadással állíthatjuk elő. Ha meg akarjuk akadályozni, hogy egy továbbadott engedélypéldány segítségével a fogadó tartomány visszavonást kezdeményezhessen, továbbadás előtt a másolatban beállítjuk a ZÁR bitet. Ez megakadályozza az engedélypéldány bármilyen módosítását, illetve a visszavonásban való felhasználását.



9.19 ábra.



9.20 ábra.

10. Felhasználóazonosítás

Az előző fejezetekben a szigorúan vett algoritmikus védelem minden kérdését megtárgyaltuk. Mindössze egyetlen feladat maradt megoldatlan: a felhasználóhoz a rendszerbe lépéskor egy olyan kezdeti ügynököt kell rendelni, amely a rendszer szempontjából a felhasználóval minden tekintetben azonosnak vehető. Mivel a felhasználó a számítógépes rendszerhez viszonyítva külső „elem”, a felhasználóazonosítás nem oldható meg kizárólag az algoritmusvédelmen belül, az ügyviteli védelem segítsége is szükséges hozzá. A cél az, hogy a felhasználóazonosítás ügyviteli védelmet igénylő mozzanata lehetőleg kezdeti és egyszeri, illetve ritkán ismétlődő legyen: a felhasználóazonosítás ismétlődő részét már teljes egészében az algoritmikus védelem végezhesse.

A felhasználó valamilyen általános célú vagy speciális terminálon keresztül lép a rendszerbe (10.1 ábra), amelyet adatátviteli vonal csatlakoztat. A felhasználó azonosítását az azonosító adatokat tartalmazó adatbázisra támaszkodva külön alrendszer végzi.

Az azonosság megállapítása alapulhat

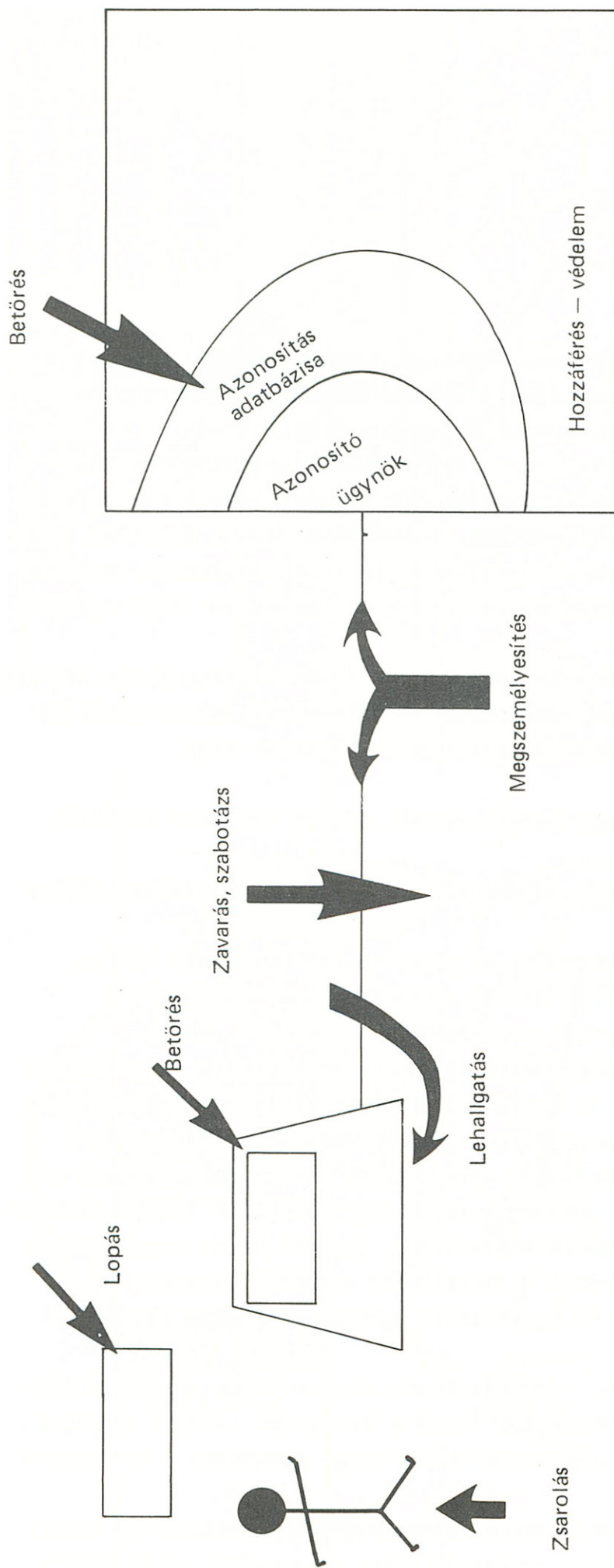
- a felhasználó személyes jellemzőin (ujj- vagy hanglenyomat, arckép helyszíni felvétele stb.),
- adatokon, információkon, amelyeket csak a felhasználó ismer (jelszó, személyes életrajzi adatok stb.) és
- a felhasználó tulajdonában lévő egyedi azonosító tárgy(ak)on (pl. azonosító kártya, amelyre egyediséget biztosító adatok vannak felírva), továbbá
- természetesen a fenti módszerek valamilyen kombinációján.

Bármelyik módszerrel történjen is az azonosítás, lényegét tekintve összehasonlításon alapul. Az azonosításhoz benyújtott (illetve az azokból kinyerhető) adatoknak, információknak meg kell egyezniük a felhasználóról a rendszerbe felvételekor eltárolt megfelelő adatokkal, információkkal. Ez utóbbiak hitelességét, védelmét és titkosságát a felvétel pillanatában az ügyviteli védelmi rendszer (pl. személyes találkozás a rendszer meghatalmazott képviselőjével, megfelelő igazoló iratok kölcsönös bemutatása, kezdeti jelszó megállapítása stb.), később pedig az algoritmikus (pontosabban a hozzáférés) védelem biztosítja.

Az azonosítás rendszerében a 10.1 ábrán is láthatóan számos lehetséges behatolási pontot fedezhetünk fel.

1. Egy felhasználó azonosító adatainak, információinak, azonosító tárgyának eltulajdonítása természetesen a legteljesebb sikert hozhatja. Ezzel a behatolási móddal szemben az algoritmikus védelem tehetetlen, mert a megszemélyesítés **teljes egészében a rendszeren kívül** történik.

2. Az azonosítási folyamat védelmében az egyik legérzékenyebb láncszem az azonosításban részt vevő terminál. Ebben ugyanis az azonosító adat, információ nyílt, illetve



10.1 ábra.

közvetlenül felhasználható formában van jelen. A terminál védelmének az alkalmazás környezeti feltételeitől függően számos módja lehetséges.

- Ha a terminál használata adminisztratív eszközökkel nyomon követhető, az ügyviteli védelem teljes egészében megoldhatja az azonosítás problémáit, hiszen a felhasználó személye ismert, és azt nem maga a felhasználó, hanem a védelmi rendszer egy megbízható alkalmazottja közli a számítógépes rendszerrel.
- Sokkal gyakoribb azonban, hogy a terminál fizikailag védett: felépítése lehetetlenné teszi szerkezetének megváltoztatását és a benne tárolt adatok eltulajdonítását vagy módosítását. Ezt részben ellenálló felépítése és burkolata, másrészt az érzékeny információkat felnyitáskor automatikusan megsemmisítő belső szerkezet biztosítja. (Az automatikus bankterminálok például az épület falán belül helyezkednek el, kívülről csak az azonosító kártyát fogadó nyílásuk és esetleg egy ellenálló felépítésű billentyűzet érhető el.)
- A jelenleg használatban lévő rendszerek többségében a fenti két megoldás együttes alkalmazása biztosítja a terminál védelmét.
- A jövő útja azonban az ún. **intelligens kártyák** használata. Ennek lényege az, hogy az azonosítási algoritmusnak azt a részét, amely érzékeny adatokat használ, egy félvezető morzsányi számítógépben végezzük, amelyet befoglalunk az azonosító kártyába. A kártyát fogadó terminálban már csak olyan adatok érhetőek el, amelyek azonosításra közvetlenül nem használhatók fel. Az intelligens kártya nem utópisztikus elképzelés: francia gyártók már (1982-től) forgalmaznak ilyeneket, elsősorban a bankügyleteket automatizáló rendszerek számára.

3. A terminál-számítógép kapcsolatba nem csupán az azonosító információ megismerésével lehet behatolni. Külön óvintézkedés nélkül előfordulhat, hogy a behatoló a megszerzett és megfejt(het)etlen adatok visszajátszásával is megszemélyesíthet egy felhasználót. Felhasználóazonosítás közben tehát mindig kötelező a párbeszéd-hitelesítés (lásd 7.1 fejezet). A számítógép pl. az azonosítás kezdetén egy véletlenszámot vagy rendszeróra-tartalmat küldhet, amellyel az azonosító terminál az azonosító információt (természetesen megfordítható módon) az alkalomra egyedivé teheti. Az azonosító rendszer ezt a módosítást eltávolíthatja, a behatoló azonban egy korábban lehallgatott azonosító párbeszédet észrevétlenül nem játszhat vissza.

4. Végül, de nem utolsósorban, a felhasználóazonosító rendszerben tárolt információk megszerzése, módosítása is sikeres megszemélyesítéshez vezethet. Az azonosítási láncnak ez az eleme különösen sok behatolási kísérletnek lehet kitéve, mert a rendszer üzemeltetői köre jelentős részének van több-kevesebb hozzáférési lehetősége a rendszer alapadataihoz.

Az algoritmikus védelem hatáskörébe elsősorban e két utóbbi behatolási terület védelme tartozik, és az adatkapcsolaton a rejtjelezést, a számítógépes rendszeren belül pedig főként a hozzáférés-védelem megismert módszereit alkalmazzák. A felhasználóazonosításnak van azonban néhány olyan jellegzetessége, amely további feladatokat jelent, illetve egyéb megoldásokat is lehetővé tesz. A fejezet hátralévő két pontjában ezekkel a kérdésekkel foglalkozunk.

10.1 Jelszóvédelem rejtjelezéssel

Ebben a pontban a számítógépes rendszerben tárolt azonosító adatok, információk védelmével foglalkozunk (az azonosító rendszer további részét pedig abszolút védettnek feltételezzük). Ennek a feladatnak a megoldása lényegében független a felhasználóazonosításra igénybe vett módszerektől, mivel

- az azonosító terminált mindegyik esetben számítógépes adatok hagyják el,
- az azonosítás végső soron a tárolt és a terminált elhagyó azonosító adatok összehasonlításán alapul, és végül
- egy felhasználó azonosítására szolgáló adatok időben állandóak vagy csak ritkán változnak.

Feltételezve tehát, hogy a felhasználót hitelesen azonosító adatok érkeznek a számítógépes rendszerbe, a védelem feladata az összehasonlításban részt vevő tárolt alapadatok védelme. Mivel ebből a szempontból az azonosítás pontos formája érdektelen, önkényesen a jelszón nyugvó azonosítást vesszük alapul, és **jelszóvédelemről** beszélünk.

A jelszóvédelem legegyszerűbb megvalósítása egy védett tábla (10.2 ábra), ahol minden elemnek két mezője van: az egyik a **felhasználó nyílt azonosítóját** (ID_i), a másik pedig a pillanatnyilag érvényes titkos **jelszavat** (PW_i) tartalmazza.

Rendszerbe lépéskor a felhasználó közli azonosítóját és jelszavát, az azonosító rendszer pedig a fenti **jelszótábla** segítségével ellenőrzi az azonosító valódiságát.

Az előbbi azonosító folyamat hitelessége érdekében a jelszótáblához csak az azonosító ügynök férhet hozzá, akinek megbízhatónak kell lennie. (A működését meghatározó program nem „szivárogtatja ki” vagy használja fel illegálisan a jelszótábla tartalmát.) E követelmény az azonosító program auditálásával (szemrevételezéssel való hitelesítésével) elégíthető ki, a másik feltételről már a legkorábbi rendszerekben bizonyossá vált, hogy túlságosan szigorú, illetve tarthatatlan. Egy számítógépes rendszer üzemeltetésének jellegzetességeiből következik, hogy az üzemeltető személyzet jelentős hányadának – a rendszer fenntartásának érdeke által diktált – jogos és korlátozhatatlan hozzáférése kell, hogy legyen a rendszerben tárolt minden egyes adathoz. A személyzet további része számára sem okoz különösebb nehézséget az adatállományokhoz való fizikai hozzáférés. Sőt mi több, a nyíltan tárolt jelszavakat elegendő csak megismerni ahhoz, hogy egy felhasználót megszemélyesíthessünk. A jelszavak védelmi módszerének tehát összetettebbnek kell lennie. Mielőtt azonban ennek kidolgozásába befognánk, foglalkozzunk röviden a jelszavak hitelesítő képességét fenyegető külső veszélyekkel.

Egy jelszó nemcsak egy behatoló kitartó igyekezete, hanem tulajdonosának gondat-

Jelszótábla

Azonosító	Jelszó
ID1	PW1
ID1	PW2
⋮	⋮
ID _n	PW _n

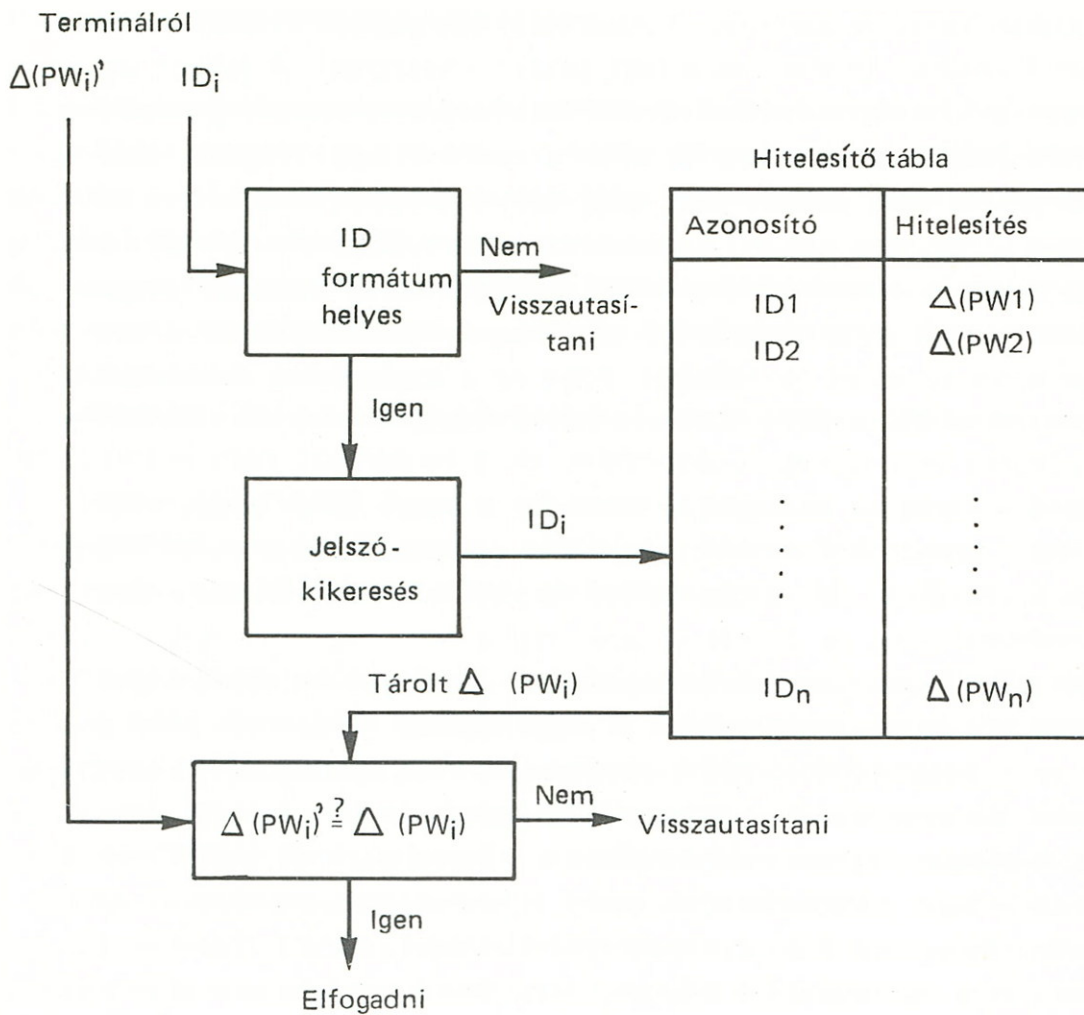
10.2 ábra.

lansága miatt is illetéktelen kezekbe kerülhet. Tulajdonosa egy jelszót – védekezésül az elfelejtése ellen – írott formában őrizhet, és a leírt jelszót elvesztheti. A jelszót ugyancsak le kell írni a rendszerbe való bejelentkezéskor is. Ha ennek nyoma marad, gondatlanság esetén az is a jelszó felfedezéséhez vezethet. Ez ellen a rendszer úgy nyújthat védelmet, hogy a bebillentyűzött jelszót nem jeleníti meg, vagy olvashatatlaná teszi. Ha a terminál billentyűzete és megjelenítő egysége közötti kapcsolatot a számítógépben záródó hurok biztosítja („ekhózás”), akkor a jelszó védelme ennek a huroknak az átmeneti megszakításával lehetséges. Ha a billentyűzet és a megjelenítő közti kapcsolat közvetlenül a terminálban jön létre, a jelszó védelmére az adhat lehetőséget, hogy ez a kapcsolat a számítógéptől jövő parancs hatására átmenetileg megszakítható (a terminál „vakon olvasó” állapotba helyezhető). A billentyűzeten leírt szöveg továbbítódik, de a terminálon nem jelenik meg. Írógépes terminálok esetében még az is elegendő lehet, ha a sorok felülírhatók (vagyis van „kocsi vissza” funkció „soremelés” nélkül). A jelszót ugyanis ilyenkor a számítógép különböző jelekkel (pl. *O, X, Z, W, M*) többszörösen és ismételten felülírhatja, amivel olvashatatlaná teszi.

A gondatlanságból előálló jelszóvesztés valószínűsége nyilvánvalóan annál nagyobb, minél hosszabb ideig van egy jelszó használatban. A számítógépes rendszerek ezért általában lehetőséget adnak arra, hogy a felhasználók jelszavaikat akár minden egyes kapcsolat során megváltoztathassák. Ez megnehezíti a behatoló helyzetét, de figyelemmel kell lennünk arra, hogy a jelszóvédelem hatásos módszereiben a jelszóváltoztatás időigényes művelet. A terminálról megvalósuló jelszómódosítás gyors és kényelmes, azonban az azonosító adatok, információk elhelyezése átlagos védelmi körülmények között történik. Ez újabb behatolási pontot jelent: az azonosító adatok, információk megszerzése születésük pillanatában, illetve módosításuk, mielőtt a rendszer adatbázisába kerülnének.

A jelszó-eltulajdonítást megkönnyíti, hogy a behatolónak olyan támpontjai is lehetnek, amelyek a jelszó „kitalálásához” vezetnek. Elsősorban is nem minden betű-szám-jel kombinációt kell figyelembe vennie mint valószínű jelszót. Emlékeztetben tarthatósági okokból a felhasználók általában számukra értelmes jelszavakat választanak, és hajlamosak arra, hogy korábbi jelszavaikat ismételten felhasználják. A jelszoválasztás mögötti logika személyenként annyira szabályos lehet, hogy a felhasználó környezete kiismerheti. (Példák: családtagok keresztnevének, a projekt nevének, a saját neve rövidítésének, a hét napjainak stb. használata.) A számítógépes rendszer ennek a veszélyforrásnak az elhárításához **jelszó-generátor** üzemeltetésével járulhat hozzá. A jelszógenerátor kérésre a felhasználó anyanyelvével azonos (tehát kiejthető, megjegyezhető) szerkezetű, egyébként azonban véletlenszerűen sorsolt jelszót kínál fel. Mivel ezek a jelszavak nem értelmesek, és nem kötődnek a felhasználó személyéhez, kitalálásuk keményebb feladat.

Visszatérve a tárolt jelszópéldány védelméhez, a 10.3 ábrán már továbbléptünk a teljes megoldás irányába. Az alapötletet az jelenti, hogy a számítógépes rendszerben a PW_i jelszó $\Delta(PW_i)$ transzformált változatát tároljuk. Ha Δ egyirányú leképezés, akkor a (jelszó) hitelesítő tábla tartalmának megismerése a jelszóra nézve nem jelent veszélyt, mert a terminálnál megadandó jelszó nem határozható meg belőle. (Emlékeztetőül: Δ egyirányú, ha $y = \Delta(x)$ ismeretében $x = \Delta^{-1}(y)$ csak x értelmezési tartományának kimerítésével és az $y? \Delta(x)$ összehasonlítás mindenkor elvégzésével határozható meg. Ne feledjük továbbá, hogy feltételeztük: a párbeszéd hitelesített, tehát a behatoló nem képes észrevétlenül belépni a terminál-számítógép kapcsolatba az adatközlő hálózaton keresztül.) A jelszó-ellenőrzés menete az ábráról nyilvánvaló, az egyetlen megtárgyalandó kérdést az egyirányú leképezés megválasztása jelenti.



10.3 ábra.

Konvencionális rejtjelezést használva a jelszót többféleképpen is védhetjük (10.4 ábra). Mindegyik esetben feltételezzük, hogy a transzformált jelszót egy kezdeti alkalommal ellenőrzött körülmények között előállítjuk, és a felhasználóazonosító rendszerben elhelyezzük.

a) Ha a terminálban rendelkezésre áll egy titkosan kezelt K kulcs (10.4.a)ábra) a

$$\Delta(PW) = \{PW\}^K$$

leképezés alkalmazható.

b) Ha PW elegendően nagy halmazból veszi értékét, akkor a közvetlenül vagy átalakítás után felhasználható a

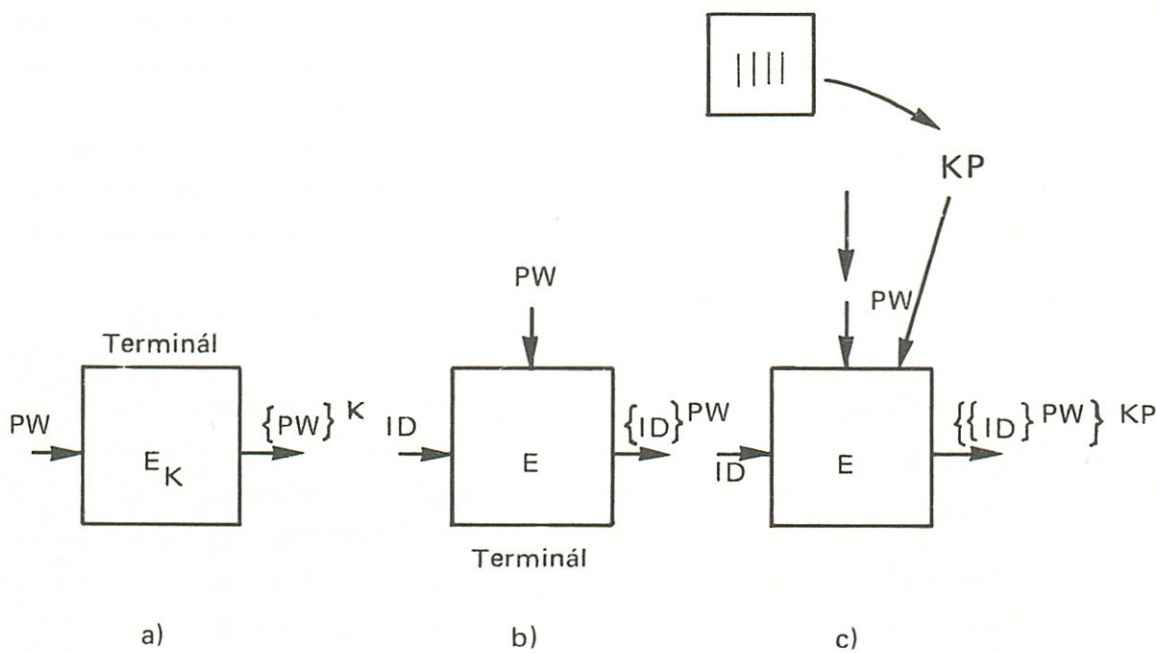
$$\Delta(PW) = \{ID\}^{PW}$$

leképezésben — ahol ID a felhasználó azonosítója — rejtjelező kulcsként (10.4 b) ábra). Ennek előnye, hogy csak a felhasználó által szolgáltatott mennyiségekre van szüksége, és ezzel megszűnik a korábbi kulcs védelmének problémája.

c) Ha az emlékezetben tartás megkönnyítésére PW egy viszonylag szűk és szabályos halmazból kerül ki, akkor további védett adatra van szükség. Ez származhat például egy mágnescsíkos azonosító kártyáról (10.4.c)ábra). Ha ezt az adatot KP -vel jelöljük, a jelszót védő leképezés például

$$\Delta(PW) = \left\{ \left\{ ID \right\}^{PW} \right\}^{KP}$$

módon is választható.



10.4 ábra.

Egyirányú leképezésekkel azonban csak a tárolt adatok megismerésén alapuló behatolás ellen tudunk védekezni. Ha a behatoló a hitelesítő táblát képes módosítani, a rendszer ismét védtelenné válik. Ha például a behatoló egy tetszőlegesen választott X érték $\Delta(x)$ transzformáltját elő tudja állítani, és ezzel a hitelesítő táblán az ID_i felhasználóhoz tartozó $\Delta(PW_i)$ -t képes helyettesíteni, akkor az X értéket mint jelszót megadva, az ID_i felhasználó nevében bejelentkezhet a rendszerbe. Ezenkívül a rendszer ID_j felhasználója egy másik ID_i felhasználó képében jelenhet meg, ha a hitelesítő táblában $\Delta(PW_i)$ -t $\Delta(PW_j)$ -re cseréli ki. Mindkét esetben a helyettesített $\Delta(PW_i)$ -t a behatoló a rendszerbe lépés után visszaállíthatja, és ezzel a leplezést elkerülheti, illetve megnehezítheti.

A hitelesítő tábla módosításával megvalósítható behatolás elleni védelmet jelentő felhasználóazonosító (jelszó-hitelesítő) folyamat a 10.5 ábrán követhető nyomon, és a hitelesítendő adat (X) mellett további két nem titkos mennyiséget használ: egy hitelesítő értéket (VP = verification pattern) és egy vizsgáló értéket (TP = test pattern). Ha a háttérben a DES rejtjelezési szabványt tételezzük fel, akkor ezek mind 64 bites mennyiségek.

X , VP és TP között két leképezés: TGEN és VGEN létesít kapcsolatot:

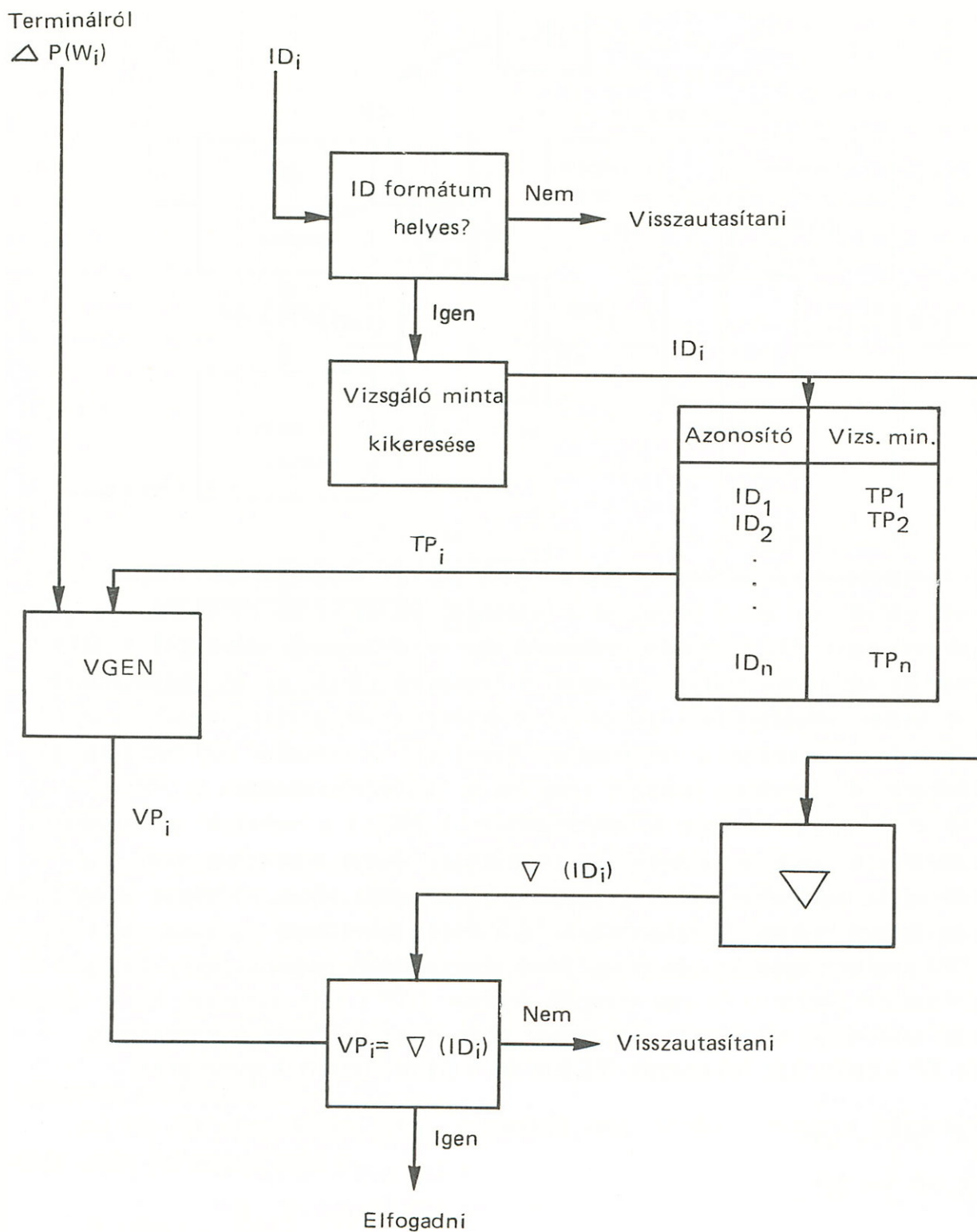
$$VGEN: \{X, TP\} \rightarrow VP.$$

$$TGEN: \{X, VP\} \rightarrow TP.$$

VGEN-ről feltételezzük továbbá, hogy ismeretében TP az (X, VP) párból semmilyen módon ki nem számítható és ki sem következtethető. VGEN tehát minden különösebb óvintézkedés nélkül és a normál munkafolyamatban felhasználható. Ezzel szemben TGEN-t csak védett körülmények között és csak titkosan szabad használni. (Pontosabban egy kezdeti fázisban, amikor a vizsgáló értéket létrehozuk.) Az X_i mennyiség hitelesítése a hozzá tartozó TP_i vizsgáló értékkel és a VGEN leképezés

$$VGEN(X_i, TP_i) \rightarrow VP_i$$

alkalmazásával, majd az eredménynek az előzetesen megadott hitelesítő értékhez való hasonlításával történik.



10.5 ábra.

Csak a helyes X_i és TP_i eredményezi a hitelesítő értékkel megegyező VP_i -t. Ha X_i -t vagy mindkettőt megváltoztatjuk, a hitelesítő értéktől eltérő eredményt kapunk.

VP_i értékének az X_i -hez kell kapcsolódnia. Ha ugyanis $VP_1 = VP_2 = \dots = VP_n = \text{állandó}$, akkor az (X_i, TP_i) párt az X_j, TP_j párral helyettesítve a rendszer „rávehető”, hogy X_j -t elfogadja X_i helyett.

A 10.5 ábrán

$$X_j = \Delta(PW_j)$$

a terminálban képződik, VP_i pedig a felhasználó azonosítójának függvénye:

$$VP_i = \nabla(ID_i)$$

ahol ∇ egyirányú leképezés, amely nem teszi lehetővé ID_i kiszámítását VP_i -ből. A leképezés alkalmazása kötelező, mert hiányában a következő behatolás lehetséges.

A behatoló választ egy PW jelszót, és egy bejelentkezési kísérlettel (miközben a terminálkimenetet lehallgatja) megszerzi $\Delta(PW)$ -t. Ha most $VP = ID$ (vagy $VP = F(ID)$ és F^{-1} az F -ből „egyszerűen” előállítható), választ egy tetszőleges TP vizsgáló értéket is, és VGEN alkalmazásával előállítja a megfelelő VP -t:

$$VGEN(\Delta(PW), TP) \rightarrow VP.$$

Ezt a VP -t (illetve $F^{-1}(VP)$ -t) azonosítóként a PW jelszó kísérletében használva, a behatoló illegálisan bejelentkezhet a rendszerbe.

Ezt a kísérletet jelentősen megnehezíti, ha ID -nek bizonyos szabályszerűségeket kell kielégítenie (pl. csak betűkből és számokból állhat), amit aztán (mint azt a 10.3 és 10.5 ábrán is tettük) a hitelesítés során ellenőrzünk. A VP -ből képzett ID ugyanis ezeket a szabályszerűségeket általában nem elégíti ki.

VP_i -t nem kell titkosan kezelni, de ellenőrzése dinamikus kell, hogy legyen, azaz nem történhet a rendszerben tárolt értékhez való hasonlítással. Ez esetben ugyanis a behatoló egy érvényes X -hez és tetszőleges TP -hez VGEN-nel előállíthatja a megfelelő VP -t, és az eredeti X -hez tartozó hitelesítő értéket a táblában erre lecserélheti. (Feltételezésünk szerint a behatoló módosítani képes az azonosító rendszer tábláit.)

A TGEN és VGEN transzformációk nyilvános kulcsú rejtjelezés esetén például

$$\begin{aligned} VGEN(X_i, TP_i) &\equiv \{X_i \oplus TP_i\}^{PK_i} \\ TGEN(X_i, VP_i) &\equiv \{VP_i\}^{SK_i} \oplus X_i \end{aligned}$$

lehetnek, ahol PK_i , illetve SK_i az ID_i felhasználó nyilvános, illetve rejtett kulcsa.

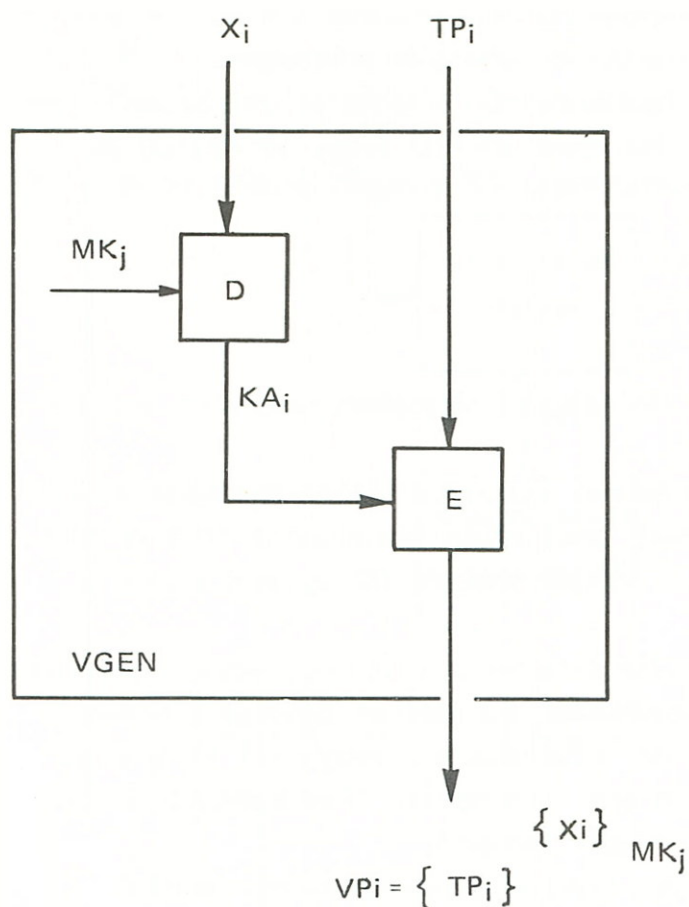
Konvencionális rejtjelezés esetén VGEN és TGEN a gazdarendszer szolgáltatása lehet, és a gazdagép mesterkulcsának egy MK_j változatát használhatják (lásd 6.1 fejezet):

$$\begin{aligned} VGEN(X_i, TP_i) &\equiv \{TP_i\}^{KA_i} \\ TGEN(X_i, VP_i) &\equiv \{VP_i\}^{KA_i} \end{aligned}$$

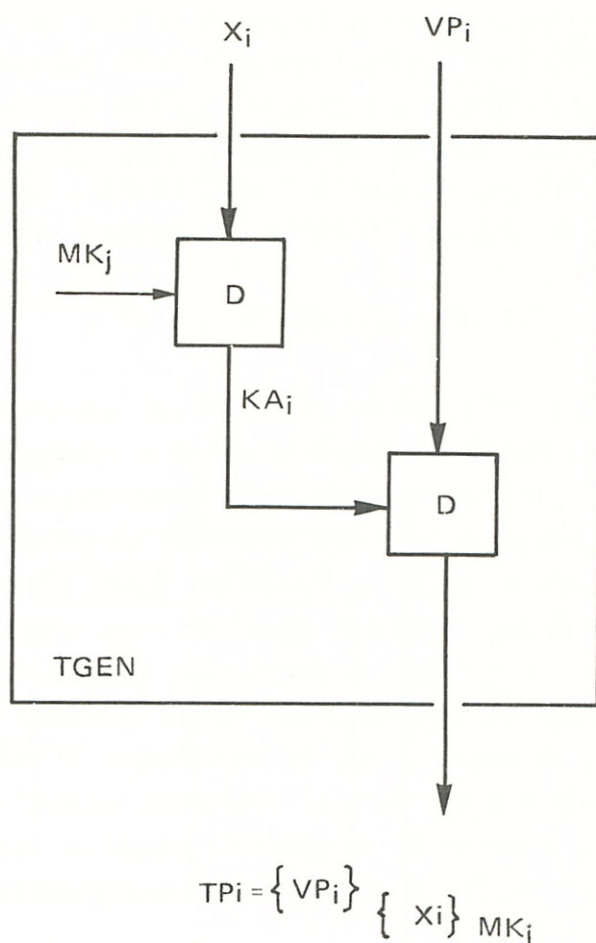
(lásd 10.6 ábra), ahol

$$KA_i = \{X_i\}^{MK_j}$$

Mivel jelszó-hitelesítésre a gazdagép-mesterkulcs egyedi változatát használjuk, a gazdarendszerben elérhető rejtjelezési transzformációk sem egyenként, sem együttesen nem nyújtanak segítséget az ismertetett hitelesítő eljárás meghamisításához.



10.6 ábra.



10.2 Jelszóvédelem jelszósorozattal

A jelszavak védelmének egyik nyilvánvaló módja, hogy minden alkalommal más és más jelszót használunk bejelentkezéskor. Ehhez természetesen az azonosító rendszernek is ismernie kell a jelszósorozatot, hogy az éppen használt jelszó hitelességet ellenőrizni tudja. Legyen a jelszósorozat

$$X_1, X_2, \dots, X_{1000}$$

és X az i -edik bejelentkezéskor használt jelszó. (Természetesen 1000 mint a sorozat hossza teljes mértékben önkényes. Mindenesetre elegendően nagynak látszik ahhoz, hogy ne kelljen túlságosan gyakran új sorozatot választani. Feltételezzük továbbá, hogy ugyanakkor elegendően kis szám ahhoz, hogy jelszavak 1000 egyirányú transzformációja elfogadható időn belül végrehajtható legyen.) Az azonosító rendszer a jelszavak hitelesítésére az

$$Y_1, Y_2, \dots, Y_{1000}$$

sorozatot használja, amelyben

$$Y_i = \Delta(X_i)$$

és Δ egyirányú leképezés. Az Y_i -knek mind különbözőeknek kell lenniük, hogy a behato-

ló ne használhasson fel egy korábban megszerzett jelszót. (Feltételezzük, hogy a behatoló ismeri Y_i sorozatot.)

Az X_{i+1} jelszó megválasztásának két nyilvánvaló módja is van:

a) Az összes X_i -t előre meghatározzuk, és az azonosító rendszer a teljes Y_i sorozatot tárolja.

b) A felhasználó Y_{i+1} -t az i -edik kapcsolat során küldi el a rendszernek, miután X_i segítségével bejelentkezett.

Egyik mód sem kielégítő minden tekintetben. Az elsőnél mind a felhasználónak, mind az azonosító rendszernek hosszú időn át 1000 adatot kell tárolnia. A másik viszont „törékeny”, adatátviteli hiba vagy a behatoló megakadályozhatja az azonosító rendszert abban, hogy a helyes Y_{i+1} -t megismerje. Mindkét nehézségen segít az alábbi módszer.

Legyen az i -edik jelszó

$$X_i := \Delta^{1000-i}(X)$$

valamilyen rögzített X mellett. (Δ^n a Δ leképezés n -szeri egymás utáni alkalmazását jelöli.) A jelszósorozat tehát

$$\Delta^{999}(X), \dots, \Delta(\Delta(\Delta(X))), \Delta(\Delta(X)), \Delta(X), X.$$

Az azonosító rendszerben felhasznált hitelesítő sorozat viszont ($Y_i = \Delta(X_i)$ miatt)

$$\Delta^{1000}(X), \dots, \Delta^4(X), \Delta(\Delta(\Delta(X))), \Delta(\Delta(X)), \Delta(X).$$

Ezek az értékek mind különböznek, ellenkező esetben Δ egyirányúsága nem teljesül. Ha pl. $Y' = \Delta^{987}(X) = \Delta^{123}(X)$ lenne, akkor feltételezésünk szerint $X' = \Delta^{986}(X) = \Delta^{863}(\Delta^{123}(X))$ elfogadható ráfordítással előállítható. Mivel $\Delta^{987}(X) = \Delta(\Delta^{986}(X))$, ez azt jelenti, hogy adott Y' -höz kiszámítható egy X' , amellyel $Y' = \Delta(X')$; ez viszont ellentmond Δ egyirányúságának.

A jelszósorozat definíciójából következik, hogy $i > 1$ -re

$$Y_i = X_{i-1},$$

azaz mindegyik jelszó egyben a következő jelszó hitelesítő értéke is. Kezdetként tehát elegendő az $Y_1 = \Delta^{1000}(X)$ értéket tudatni az azonosító rendszerrel, később ezt mindig a felhasználó által utolsónak megadott jelszóval lehet helyettesíteni. (A rendszernek tehát mindig csak egyetlen értéket kell tárolnia.)

Hogy a módszer védettségét belássuk, tételezzük fel például, hogy az első 987 jelszó, $\Delta^{999}(X), \dots, \Delta^{13}(X)$, ismerete a behatoló számára lehetővé teszi a következő jelszó, $\Delta^{12}(X)$ meghatározását. Ebből azonban következik, hogy ha adott $Y' = \Delta^{13}(X)$, akkor először $\Delta^{14}(X), \dots, \Delta^{999}(X)$, majd feltételezésünk szerint ezek ismeretében $X' = \Delta^{12}(X)$ meghatározható. Mivel azonban $Y' = \Delta(X')$, ez ellentmond Δ egyirányúságának. Mivel a jelszósorozatot előre rögzítettük, a behatoló semmilyen módon nem képes magából a már megismert sorozatból egy későbbi jelszót meghatározni.

A most ismerttetett módszernek van egy fontos hibatűró tulajdonsága is: ha lépéskülönbség keletkezik a felhasználó és a rendszer között — a felhasználó X_i -t küldi

és a rendszer Y_k -t kísérli meg felhasználni hitelesítésre, miközben $j \neq k$, akkor ez felfedhető, ha mind a jelszóra, mind a hitelesítő értékre ismételten alkalmazzuk Δ -t mindaddig, amíg egyezést nem kapunk. Például: ha a felhasználó X_j -t küldi, és a rendszer Y_{j+3} -t használja hitelesítésre, akkor ez felfedhető, mert $X_j = \Delta^2(Y_{j+3})$. A helyzet fordított, ha a jelszó indexe a nagyobb. A rendszer elfogadhatja X -t, ha $j > k$, új értéket kérhet, ha $j < k$. (Ez utóbbi esetben elképzelhető ugyanis, hogy a behatoló kísérletezik.)

Ez a tulajdonság lehetővé teszi annak megakadályozását, hogy a behatoló hasznot húzzon egy esetleges „rendszerösszeomlásból”. A rendszer újraindítása ilyen helyzetben ugyanis általában megköveteli, hogy egy olyan korábbi helyzetet állítsunk vissza, amelyben a rendszer egy, a felhasználó által már egyszer elküldött jelszót vár. Tegyük fel például, hogy a behatoló rutinszerűen feljegyzést vezet a lehallgatott jelszavakról, és a rendszerösszeomlás azután következik be, hogy a felhasználó az X_{374} jelszót használta. Ha az azonosító rendszert abba az állapotba állítanánk vissza, amelyben az Y_{374} -gyel hitelesíthető jelszót vár, a behatoló feljegyzései alapján megsemmisíthetné a felhasználót, mert már lehallgatta X_{374} -t.

Az ismertetett módszer alkalmazásakor nem kell visszalépni egy, már korábban használt jelszóig. Helyette az azonosító rendszert előre „ugrattathatjuk”. Tegyük fel például, hogy a teljes rendszert egy olyan előző állapotba állítottuk vissza, amelyben az azonosító rendszer $Y_{374} = \Delta^{626}(X)$ -t használta jelszóhitelesítésre. Tegyük fel továbbá, hogy egy felhasználó óránként egynél több bejelentkezést sohasem hajt végre. Ha az összeomlás és az újraindítás között kevesebb, mint két óra telik el, biztosak lehetünk abban, hogy a felhasználó nem juthatott túl az X_{375} jelszón. Az azonosító rendszer ezért felkérheti a felhasználót, hogy X_{376} -t küldje el, mivel azt csak a felhasználó képes előállítani. Ezt a jelszót viszont a rendszer hitelesíteni tudja, mert $Y_{374} = \Delta^3(X_{376})$. A behatoló tehát még rendszerösszeomlás esetén sem tudja felhasználni a korábban megismert jelszavakat.

- [1] FEISTEL, H. — NOTZ, W. A. — SMITH, J. L.: Some Cryptographic Techniques For Machine-to-Machine Data Communications, Proc. IEEE 1975. nov. 1545—1554.
- [2] DIFFIE, W. — HELLMAN, M. E.: Privacy and Authentication; An Introduction to Cryptography. Proc. IEEE 1979. márc. 397—427.
- [3] LEMPEL, A.: Cryptology in transition Computing Survey 1979. dec. 285—303.
- [4] SIMMONS, G. J.: Symmetric and assymetric encryption Computing Surveys 1979. dec. 305—330.
- [5] STAHL, F. A.: A homophonic cipher for computational cryptography. Proc. AFIPS NCC 1973. 565—568.
- [6] MELLEN, G. E.: Cryptology, computers and common sense. Proc. AFIPS NCC 1973. 569—579.
- [7] MEYER, C. H.: Design consideration for cryptography. Proc. AFIP NCC 1973. 603—606.
- [8] PELEG, SH. — ROSENFELD, A.: Breaking Substitution Ciphers Using a Relaxation Algorithm. CACM 1979. nov. 598—605.
- [9] GILBERT, E. N. — MacWILLIAMS, F. J. — SLOANE, J. A.: Codes Which Detect Deception. BSTJ 1974. márc. 405—424.
- [10] THE DATA ENCRYPTION STANDARD. FIPS—PUB 46, National Bureau of Standards USA.
- [11] MEYER, C. H.: Ciphertext plaintext and ciphertext/key dependence vs number of rounds for the data encryption standard. Proc. AFIPS NCC 1978. 1119—1126.
- [12] DIFFIE, W. — HELLMAN, M. E.: Exhaustive Cryptanalysis of the NBS Data Encryption Standard IEEE Computer 1977. jun. 74—84.
- [13] HELLMAN, M. E.: A Cryptanalytic Time-Memory Trade-Off. IEEE Trans IT—26 1980. jul. 401—406.
- [14] MERKLE, R. C. — HELLMAN, M. E.: On the Security of Multiple Encryption. CACM 1981. jul. 465—467.
- [15] HELLMAN, M. E.: An Extension of the Shannon Theory Approach to Cryptography. IEEE. Trans. IT—23 1977. máj. 289—294.
- [16] SUGARMAN, R.: On foiling computer crime. IEEE Spectrum 1979. jul. 31—41.
- [17] KONHEIM, A. G. ÉS MÁSON: The IPS cryptographic programs. IBM System Journal 1980. 2. sz. 253—283.
- [18] CHAUM, D. L.: Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. CACM 1981. febr. 84—88.
- [19] KLINE, CH. S. — POPEK, G. J.: Public key vs. conventional key encryption. Proc. AFIPS NCC 1979. 831—837.
- [20] DENNING, D. E.: Secure Personal Computing in an Insecure Network. CACM 1979. aug. 476—482.
- [21] SMID, M. E.: Integrating the Data Encryption Standard into Computer Networks. IEEE Trans COM—29 1981. jun. 762—772.
- [22] HELLMAN, M. E.: Security in communication networks. Proc. AFIPS NCC 1978. 1131—1134.
- [23] POPEK, G. J. — KLINE, CH. S.: Encryption and Secure Computer Networks Computing Surveys 1979. dec. 331—356.
- [24] FLYNN, R. — CAMPASANO, A. S.: Data dependent keys for a selective encryption terminal. Proc. AFIPS NCC 1978. 1127—1129.

- [25] MATYAS, S. M.: Digital Signatures — An Overview. *Computer Networks* 1979. 3. sz. 87–94.
- [26] DEMILLO, R. A. (EDITOR): *Foundations of Secure Computation*. Academic Press 1978.
- [27] RABIN, M. O.: Digitalized signatures in *Foundations of Secure Computation*. Academic Press 1978.
- [28] BOOTH, K. S.: Authentication of Signatures Using Public Key Encryption. *CACM* 1981. nov. 772–774.
- [29] MEIJER, H. — AKL, S.: Digital signature schemes for computer communication networks. *Proc. 7th Data Communications Symposium* 1981.
- [30] DIFFIE, W. — HELLMAN, M. E.: New Directions in Cryptography. *IEEE Trans. IT-22* 1976. nov. 644–654.
- [31] DIFFIE, W. — HELLMAN, M. E.: Multiuser cryptographic techniques. *Proc. AFIPS NCC* 1976. 109–123.
- [32] RIVEST, R. L. — SHAMIR, A. — ADLEMAN, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystem. *CACM* 1978. feb. 120–126.
- [33] WILLIAMS, H. C.: A Modification of the RSA Public-Key Encryption Procedure. *IEEE Trans IT-26* 1980. nov. 726–729.
- [34] POHLING, S. C. — HELLMAN, M. E.: An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans IT-24* 1978. jan. 106–110.
- [35] MERKLE, R. C. — HELLMAN, M. E.: Hiding Information and Signatures in Trapdoor Knapsacks. *IEEE Trans IT-24* 1978. szept. 525–530.
- [36] SHAMIR, A. — ZIPPEL, R. E.: On the Security of the Merkle–Hellman Cryptographic Scheme. *IEEE Trans IT-26* 1980. máj. 339–340.
- [37] ARAZI, B.: A Trapdoor Multiple Mapping. *IEEE Trans. IT-26* 1980. jan. 100–105.
- [38] HENRY, P. S.: Fast Decryption Algorithm for the Knapsack Cryptographic System. *BSTJ* 1981. máj.–jun. 767–773.
- [39] MERKLE, R. C.: Protocols for Public Key Cryptosystems. *IEEE* 1980. Symposium on Security and Privacy, 122–134.
- [40] LUCCIO, F. — MAZZONE, S.: A cryptosystem for multiple communication Information Processing Letters 1980. jul. 5. 180–183.
- [41] MERKLE, R. C.: Secure Communication over Insecure Channels *CACM* 1978. ápr. 294–299.
- [42] INGEMARSSON, I. — WONG, C. K.: Encryption and Authentication in On–Board Processing Satellite Communication Systems. *IEEE Trans COM-29* 1981. nov. 1684–1687.
- [43] DENNING, D. E. — SCHNEIDER, F. B.: The master key problem. *IEEE* 1980. Symposium of Security and Privacy, 103–107.
- [44] NEEDHAM, R. M. — SCHROEDER, M. D.: Using Encryption for Authentication in Large Networks of Computers. *CACM* 1978. dec. 993–999.
- [45] WEGMAN, M. N. — CARTER, J. L.: New Hash Functions and Their Use in Authentication and Set Equality. *J. of Computer and System Sciences* 1981. 265–279.
- [46] DENNIND, D. E. — SACCO, G. M.: Timestamps in Key Distribution Protocols. *CACM* 1981. aug. 533–536.
- [47] KENT, S. T.: Security Requirements and Protocols for a Broadcast Scenario. *IEEE Trans COM-29* 1981. jun. 778–786.
- [48] McELIECE, R. J. — SARWATE, D. V.: On Sharing Secrets an Reed–Solomon Codes. *CACM* 1981. szept. 583–584.
- [49] EHRSAM ET AL.: A cryptographic key management scheme for implementing the Data Encryption Standard. *IBM System Journal* 1978. 2.sz. 106–125.
- [50] MATYAS, S. M. — MEYER, C. H.: Generation, distribution and installation of cryptographic keys. *IBM System Journal* 1978. 2.sz. 126–137.
- [51] SHAMIR, A.: How to share a secret. *CACM* 1979. nov. 612–613.
- [52] BLAKLEY, G. R.: Safeguarding cryptographic keys. *Proc. AFIPS NCC* 1979. 313–317.
- [53] EVANS, A. — KANTOROWITZ, W.: A user authentication scheme not requiring secrecy in the computer. *CACM* 1974. aug. 437–441.
- [54] PURDY, G. B.: A high security log-in procedure. *CACM* 1974. aug. 442–445.
- [55] LAMPORT, I.: Password authentication with insecure communication. *CACM* 1981. nov. 770–772.
- [56] LENNON, R. E. — MATYAS, S. M. — MEYER, C. H.: Cryptographic authentication of time-invariant Quantities. *IEEE Trans. COM-29* 1981. jun. 773–777.

- [57] MEYER, C. H. — MATYAS, S. M.: Some cryptographic principles of authentication in electronic funds transfer system. Proc. 7th Datacommunication Symposium (IEEE), 1981.
- [58] MEYER, C. H. — MATYAS, S. M. — LENNON, R. E.: Required cryptographic authentication criteria for electronic fund transfer systems. Proc. 1981. IEEE Symposium on Security and Privacy
- [59] GIFFORD, D. K.: Cryptographic sealing for information secrecy and authentication. CACM 1982. ápr. 274–286.
- [60] GUDER, E.: The design of a cryptographic based secure file system. IEEE Trans SE—6 1980. szept. 411–420.
- [61] DAVIDA, G. I. — WELLS, D. L. — KAM, J. B.: A database encryption system with subkeys ACM Trans. Data Base Systems 1981. jun. 312–328.
- [62] GLIGOR, V. D.: Review and revocation of access privileges distributed through capabilities. IEEE Trans. SE—5 1979. nov. 575–586.
- [63] REDELL, D. D.: Naming and protection in expandable operating systems. MAC TR—140, Massachusetts Institute of Technology
- [64] DENNING, D. E. — DENNING, P. J.: Data Security. Computing Survey 1979. szept. 227–249.
- [65] SALTZER, J. H. — SCHROEDER, M. D.: The protection of information in computer systems. Proc. IEEE 1975. szept. 1278–1308.
- [66] COHEN, E. — JEFFERSON, D.: Protection in the HYDRA operating system Proc. 5th Symposium on Operating System Principles 1975.
- [67] DENNING, P. J. Fault tolerant operating systems Computing Surveys 1976. dec. 359–389.
- [68] MINSKY, N.: An operation-control scheme for authorization in computer systems. International Journal of Computer and Information Sciences 1978. 2.sz. 157–191.
- [69] MINSKY, N.: The principle of attenuation of privileges and its ramifications in Foundation of Secure Computation. Academic Press 1978.
- [70] LINDEN, TH. A.: Operating System structures to support security and reliable software. Computing Surveys 1976. dec. 409–445.
- [71] STIEGLER, H. G.: A structure for access control lists. SOFTWARE — Practice and Experience 1979. 813–819.
- [72] FABRY, R. S.: Capability-based Addressing. CACM 1974. jul. 403–412.

Йожеф Нимет

ЗАЩИТА ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ И В СИСТЕМАХ НА БАЗЕ ЭВМ

По мере дальнейшего распространения систем на базе ЭВМ, деятельность и многочисленные услуги нашей повседневной жизни становятся все более зависимыми от ЭВМ. Именно поэтому, все более важное место занимает вопрос предотвращения нелегального использования таких систем.

Исходящая из этого задача разлагается на множество подзадач, среди которых в данной книге рассматриваются те, которые относятся к кругу алгоритмической защиты. Книга подробно знакомит читателей с основным кругом тем по идентификации пользователей, применению секретных ключей и защиты доступа. При этом, в описании фигурируют как и элементы процедуры, так и системно-технические решения.

József Németh:

DATA PROTECTION IN COMPUTER AND TELECOMMUNICATIONS SYSTEMS

With the rapid proliferation of computer based systems we are rapidly becoming completely dependent on computers in many activities and services of our everyday life. It is thus increasingly important to prevent the misuse and abuse of such systems. This intrinsically complex task can be broken down into several levels and subsystems of which those belonging to algorithmic protection are discussed in this book. User authentication, cryptographical protection and access control are the major topics treated with detail. Both procedural elements and system solutions are presented.

Kiadja a Számítástechnika-alkalmazási Vállalat
Felelős kiadó: Juhász János vezérigazgató

Felelős szerkesztő: Huba Zoltánné
Műszaki vezető: Molnár Zoltán
Műszaki szerkesztő: G. Müller Zsuzsa
A fedelet tervezte: Hörömpő Gabriella
Az ábrákat rajzolta: Kapusi Zoltán
Megjelent: 14 (A/5) ív terjedelemben

Készült a SZÁMALK reprográfiai üzemében
Nyomdai megrendelés törzsszáma: 84/295
Budapest, 1984

Ára: 69,— Ft